
# Managing Flash Media Server

**Acknowledgments**

Project Management: Suzanne Smith

Writing: John Norton, Suzanne Smith

Editing: Geta Carlson, Evelyn Eldridge, Mary Ferguson, Lisa Stanziano, Anne Szabla

Production Management: Adam Barnett

Media Design and Production: Aaron Begley, Paul Benkman, John Francis, Mario Reynoso

First Edition: October 2005

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103

# Contents

# About This Manual

Macromedia Flash Media Server 2 enables one-to-one, one-to-many, many-to-one, and many-to-many communication in real time between applications created in Macromedia Flash 8. Developers create these applications using ActionScript, a scripting language based on the same standard used by the JavaScript language.

Flash Media Server communicates with Macromedia Flash Player using the Real-Time Messaging Protocol (RTMP), an unencrypted TCP/IP protocol designed for high-speed transmission of audio, video, and data messages. You can also administer the server over HTTP. You can use the same server administration application programming interface (API) over HTTP as you would over RTMP. By passing command strings and arguments to the URL of your Flash Media Server, you can interact with the server to retrieve information or modify the server configuration. This API is described in detail in the *Server Management ActionScript Language Reference*, included with Flash Media Server.

This manual describes how to configure and manage Flash Media Server to support media applications deployed on a variety of network configurations. The rest of this chapter provides system requirements, describes the Flash Media Server documentation, and lists additional resources.

Flash Media Server enables applications to communicate with other servers. This manual does not discuss web server and application server management or server operating system setup.

## Intended audience

This manual is aimed at system administrators who will configure and manage Flash Media Server to support media applications. You should already be familiar with basic network infrastructure and security. You should also have some familiarity with client-server application models, XML, and JavaScript.

# System requirements

Flash Media Server can be deployed on the following systems:

- Microsoft Windows 2000 Server or Windows 2003 Server - Standard Edition running on a Pentium III 1-GHz processor or faster (Dual Pentium 4 or faster recommended).

  Windows XP is acceptable for developing and testing applications.
- Linux Red Hat Enterprise Version 3.0 and Linux Red Hat Enterprise Version 4.0 running on a Pentium III 1-GHz processor or faster (Dual Pentium 4 or faster recommended).

Your deployment system for Flash Media Server applications also requires the following:

- Minimum of 512 MB of available RAM.
- 50 MB of available disk space.
- CD-ROM drive for installation.

Depending on your applications, your requirements might be greater than outlined here.

If you install Flash Media Server on a Linux computer, you'll need Flash 8 installed on a separate Windows or Macintosh computer to develop your Flash applications. You'll also need Macromedia Flash Player for Windows or Macintosh and a web browser to run the sample applications.

# About the Flash Media Server documentation

All Flash Media Server documents are available in PDF format (viewable and printable with Adobe Acrobat Reader at www.adobe.com/products/acrobat/) and as Flash help. The PDFs are available on the product CD or can be downloaded from the Macromedia website. To view the help, open the Welcome page (in Windows, Start > Programs > Macromedia > Flash Media Server > Welcome) or in Flash 8, select Help > Using Flash and then select a Flash Media Server topic from the Table of Contents.

Tutorials on how to build simple applications that demonstrate important individual concepts are included with the server. You can access these tutorials on Windows by selecting Start > Programs > Macromedia > Flash Media Server > Welcome. Click on Welcome and then Sample Applications. A link to the tutorials appears on the page that is displayed.

# Typographical conventions

The following typographical conventions are used in this book:

- `Code font` indicates ActionScript statements, XML tag and attribute names, and literal text used in examples.
- *Italic* indicates placeholder elements in code or paths. For example, /settings/*myPrinter*/ means that you should specify your own location for *myPrinter*.
- Directory paths are written with backslashes (\) for servers running on Microsoft Windows systems. If you are running Flash Media Server on a UNIX system, replace the backslashes with forward slashes (/).

# Additional resources

The Flash Media Server documentation was written before the code in the product was complete. Therefore, there may be discrepancies between the final implementation of the product's features and how they are documented in this manual. For a list of known discrepancies, see the documentation update (www.macromedia.com/go/flashmediaserver_documentation_update_en) in the Flash Support Center (www.macromedia.com/go/flashmediaserver_support_en).

The Flash Support Center is updated regularly with the latest information on Flash and Flash Media Server, as well as advice from expert users, advanced topics, examples, tips, and other updates.

CHAPTER 1

# Managing the Server

1

As a Flash Media Server administrator, you'll need to perform several administrative tasks after the server is installed. This chapter describes how Macromedia Flash Media Server is configured when you first install it, how to set up additional administrators, and how to monitor the server's activity. For many of these tasks, you'll use the management console that was installed with the server. This chapter describes the management console in detail.

## Basic server settings

The server is installed with a set of configuration files in XML format. These files define a default server adaptor, a default applications directory, default server administrators, and default settings for application behavior.

The default server adaptor uses port 1935, the number assigned to Flash Media Server by the Internet Assigned Numbers Authority (www.iana.org). Although you can use any port number, this increases the risk of conflicting with another application that may be assigned to the same port; for example, if you configure the server to use port 80 to support HTTP tunneling, the server might not run both a web server and the Flash Media Server bound to port 80. Applications must be authored to connect to the same port the server is using, in the NetConnection.connect statement. Be sure the port is not blocked by a firewall.

The server is preconfigured with one adaptor containing one virtual host. The virtual host is equivalent to a domain name. The default applications directory for the default virtual host is the applications directory in the Flash Media Server 2 directory. You can view this location by looking at the value for the `AppsDir` tag in the vhost.xml file. This directory is where the server will look for application subdirectories at startup; you must place an application subdirectory here for each client application that you plan to connect to the server, and the client subdirectory must have the same name as the client application. The presence of the application subdirectory registers the application with the server.

You can configure Flash Media Server 2 as an origin or edge server, configure adaptors and virtual hosts and change the location of the applications directory by editing the server's configuration files and creating directories in the server's conf directory. For more information, see Chapter 2, "Deploying Flash Media Server."

The default server administrator has the user name and password you chose during the Flash Media Server installation, and is defined in the Users.xml configuration file. The server administrator can connect to the Admin service with the management console and perform a variety of server administration tasks, including shutting down the server and disconnecting client applications. In the nomenclature of server administration, this server administrator is equivalent to the UNIX superuser.

Virtual host administrators can only perform tasks relating to the applications running on their own virtual host. There are no virtual host administrators defined when the server is first installed. Server administrators, including the default server administrator defined during installation, have access to all virtual hosts. Server administrators can add or delete virtual host administrators using the management console.

## Registering client applications

The server is configured at installation with one adaptor directory named _defaultRoot_ containing one virtual host directory named _defaultVHost_. The server defines its virtual hosts at startup by searching for directories within the adaptor directory that contain valid Vhost.xml files, such as the _defaultVHost_ directory. At the same time, the server defines each application that will be allowed to connect to a virtual host by looking for application directories inside a directory specified by the `AppsDir` tag in the Vhost.xml file (by default, the applications directory in the Flash Media Server directory).

For example, if you create an application named my_app, you must create a my_app subdirectory in the applications directory to register my_app.

You can specify the directory you want to use to store your client application directories by editing the `AppsDir` tag in the Vhost.xml file. By changing the path specified in this tag, you can locate the directory for the virtual host anywhere you want. If no directory is specified, it defaults to the virtual host directory itself.

**To edit the <AppsDir> tag in the Vhost.xml file:**

1. Locate the Vhost.xml file for the virtual host you are working with.
2. Open the file in a text editor.
3. Replace the path inside the `AppsDir` tag with the path of your choice, such as C:\Server Files\applications. Do not use quotation marks. To specify multiple directories, delimit each directory path with a semicolon.
4. Save the Vhost.xml file.

You must restart the server in order for this change to take effect.

Once you have specified the directory where you'll store your application directories, you must create a directory inside it for each client application you plan to use with that virtual host. Each client application must have a directory with the same name that the client application uses when connecting to the server. Once you have created a subdirectory for each of your applications, you can decide whether to give any of the applications their own Application.xml file. By including an Application.xml file in a client application's directory, you can give that application different settings from those defined in the virtual host's Application.xml file, which serve as the default settings for applications on the virtual host. For more information about the Application.xml file, see Chapter 2, "Deploying Flash Media Server."

## Configuring virtual hosts

With some editions of the server, you can add virtual hosts to the server's configuration. This is useful for separating sets of applications and allows you to define administrators who have access only to a specific virtual host. It is also useful if you are an Internet Service Provider who is hosting websites using Flash Media Server.

Each virtual host on the server is associated with an adaptor.

To create a new virtual host, create a new virtual host directory inside the /conf/*adaptor_name* directory in the Flash Media Server directory, for the adaptor you want to use for the new virtual host: /conf/*adaptor_name*/*virtual_host_name*. This directory must include the following configuration files:

- Vhost.xml
- Application.xml
- Users.xml
- Logger.xml

You also need to add the new virtual host to the Server.xml file and add administrative users for that virtual host.

## Deploying server-side scripts

In developing applications for Flash Media Server, you may decide to use server-side scripts to implement some of the functionality. To deploy server-side scripts, you can store them in your registered application directory for the application that uses them or in a scripts directory (which you create) within your application directory. Server-side scripts should always reside on the computer where Flash Media Server 2 is installed.

For example, you could store the main.asc file for my_app in the following directory: *installation_directory*/applications/my_app/scripts.

If you create server-side scripts that use characters that are not in the classic 8-bit ASCII character set, such as non-English characters, you must use a text editor that encodes text in UTF-8 format. Macromedia Dreamweaver can encode text in this format.

Script files that are encoded in UTF-8 format must be transferred to the server via a binary file transfer.

For more information about using server-side scripts, see *Developing Media Applications.*

## Starting and stopping the server in Windows

In Windows, Flash Media Server runs as a service; it does not appear in the Windows taskbar. Therefore, you don't shut down or restart the server as you would for other Windows applications.

You can shut down and restart the server in a few ways:

- Use the management console to connect to the server and then shut it down or halt it remotely. Only server administrators can perform these tasks.

  For more information, see "Using the management console" on page 16.
- Use the Windows Services control panel.

### To start or stop the server:

In the Start menu, select Programs > Macromedia > Flash Media Server 2> Start Service or Stop Service.

### To stop the server in the Services control panel:

1. In the Windows Start menu, select Settings > Control Panel.
2. In the Control Panels folder, double-click the Administrative Tools folder.
3. In the Administrative Tools folder, open the Services control panel.

4. In the Services list, scroll down and select Flash Media Server.
5. Click the Stop button at the top of the control panel.

   The server shuts down.

### To restart the server in the Services control panel:

1. Open the Services control panel.
2. Select Flash Media Server.
3. Click the Start button at the top of the control panel.

   The server starts up.

## Starting and stopping the server on Linux

On Linux systems, Flash Media Server is installed as a service. You start and stop the Flash Media Server service using the `fmsmgr` utility.

### To start the server on Linux:

1. Log in as a root user.
2. Change to the directory where the server is installed.
3. Open a shell window and type the following:

   ```
   fmsmgr server fms start
   ```

### To stop the server on Linux:

1. Log in as a root user.
2. Change to the directory where the server is installed.
3. Open a shell window and type the following:

   ```
   fmsmgr server fms stop
   ```

Use the `fmsmgr` utility to perform other tasks as well, such as configuring the service to start automatically when the system is started. For more information on this and other tasks, see "Using the fmsmgr utility" on page 58.

### To start the Admin service on a Linux system:

1. Log in as a root user.
2. Change to the directory where the server is installed.
3. Open a shell window and type the following:

   ```
   fmsmgr server adminserver start
   ```

For more information on fmsmgr commands, see "Using the fmsmgr utility" on page 58.

# Using the management console

The management console for the Flash Media Server 2 release has been redesigned to ease the workflow for administrators. It is also enhanced with many new features. From the graphical user interface, you can administer servers running Flash Media Server, monitor their processes, and debug their applications.

For example, a designer debugging an application wants to view the content of a particular stream. The administrator logs in to the Flash Media Administrative service through the management console, selects the application and then clicks its Streams tab. They select the desired stream from the list and can replay it within the management console by selecting the Play Streams option.

Another user debugging a different application wants to review the contents of a shared object implemented in their application and examine the properties in this shared object. The user logs into the management console and selects the application. The user moves to the Shared Objects tab and selects the shared object. The management console displays the object's data properties in the adjoining window.

The management console is a Flash application (fmsconsole.swf) that Macromedia created with public APIs (application programming interfaces). When you install Flash Media Server 2 on Windows or Linux systems, the installer places fmsconsole.html in the Flash Media Server 2*directory* [the root of the installed directory]. If you have Flash Player installed, you can monitor and control the server's activity by launching the management console and connecting to the server.

To run the management console from a computer other than the computer where the server is installed, copy fmsconsole.html and fmsconsole.swf to the other computer, or make sure that this file is in your webroot directory so it can be accessed remotely. In both cases, you'll need to make sure that the `Allow` and `Deny` tags in the Users.xml file allow the connection from the other computer's IP address.

For more information, see "Server administration over HTTP" on page 71.

With the management console you can perform the following tasks:

- Check the status of the server and the applications running on it.
- Shut down or restart the server, a virtual host, or individual client applications.
- Add and edit administrators.
- View server performance data such as client connections, bandwidth, CPU, and memory usage.

- View application logs.
- View logs of server connections and other server events.
- View streams and inspect shared object data.
- View and update the server's license key and its bandwidth and connection limits.

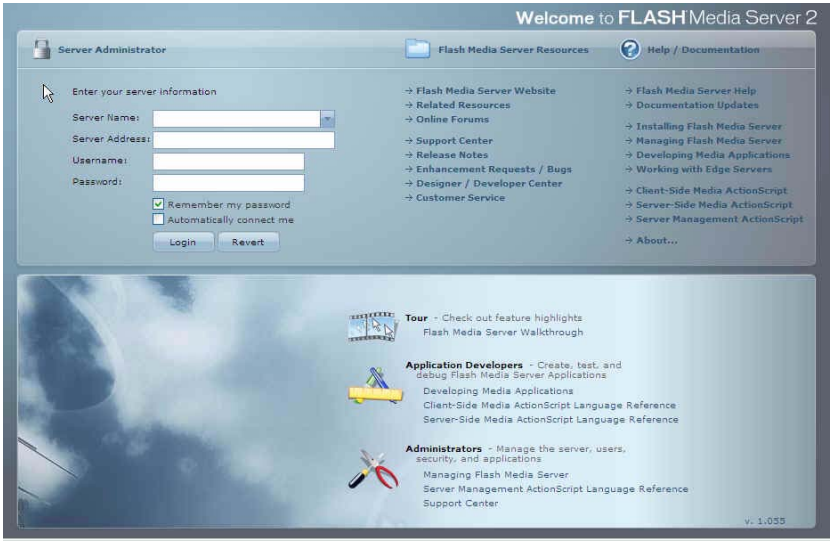
As a security feature, when you connect to the server with the management console, it actually connects to a separate Admin service that runs in parallel with the server service. The Admin service then communicates with the server to perform its administration functions. On both Linux and Windows systems, you must first explicitly start the Admin service. After the Admin service is started, authorized administrators can access the management console.

## Connecting to the management console

You can access the management console from computers running Windows, Linux or Macintosh operating systems.

### To connect to the management console:

1. In Windows, from the Windows Start menu, select Programs > Macromedia > Flash Media Server > Console. In Linux, open the fmsconsole.html file in a web browser on a computer where Flash Player is installed.



2. Enter the name and address of the server or virtual host you want to connect to.
   - You can enter *localhost*, which will refer to the computer that the management console is running on.
   - If you are connecting remotely by running the management console on another computer, enter the server's name (FMS.myCompany.com) or the IP address and port number of the server you want to connect to (12.34.56.78:1112).
   - Make sure your computer has permission to connect to the specified port on the other computer.
3. Enter the administrator's user name and password.

   Enter the name and password you entered during the Flash Media Server installation. If you've changed the administrator user name and password using the management console or manually in the Users.xml file, enter the new user name and password.

   When logging on to a virtual host that is not on the default adaptor, virtual host administrators must specify the name of the adaptor. For example, if a virtual host administrator is logging on to a virtual host on the adaptor `_secondAdaptor_`, the administrator `JLee` would enter the following information in the Name box: `_secondAdaptor_/JLee`.
4. If you want the management console to remember your login and password when you use it in the future, select the Remember My Password option.
5. If you want the management console to automatically connect to the server when you open it, select the Automatically Connect Me option.

   Click the Login button.
6. Click the Revert button to return the management console to its default settings.

   Be aware that this action will delete all saved servers, user names, and passwords from the management console. All custom resizing within the management console will be restored to the original state.

   The Revert button, however, does not affect the server.

You are now connected to the server. At this point you can go to the three main sections of the management console:

- View Applications
- Manage Users
- Manage Servers

## Setting the refresh rate

The management console provides live performance data on the server. You can use the Refresh Rate pop-up menu to control how often the information displayed on the management console is updated. The default rate is five seconds. You can also use the pop-up menu to pause refreshing the information.

## Accessing Flash Media Server resources and help

Near the top of every screen of the management console you will find two icons. Clicking the folder icon displays links to the following resources:

- Flash Media Server website
- Related resources

  These organizations provide additional educational and consulting resources for a successful implementation.

  - Online forums
  - Support Center
  - Release Notes
  - Enhancement Requests and Bugs
  - Designer/Developer Center
  - Customer Service

Clicking the question mark icon displays links to Flash Media Server online help and documentation.

# Managing applications

The View Applications panes display information about the applications that are running on the specified server or virtual host. From these panes the server or virtual hosts administrator can monitor the state of an application.

Here are some likely uses of the management console:

- A developer wants to see the statistics on how many clients are connected to the application, and view the live log, which shows the trace() statements made by the application.
- A user debugging their application wants to see what the content of a particular stream playing back is. They log in to the management console and click the View Applications panel. After choosing a particular stream and pressing Play Stream, a pop-up window appears and plays back the streaming content inside the management console.

- A user debugging another application now desires to see the contents of a shared object that they have implemented in their application. Specifically they would like to examine the properties in this shared object. Now the user logs into the management console and selects their application. After the application is selected, the user moves to the Shared Objects tab and select the shared object of choice. The object's properties are now displayed for examination in the adjoining window.


When you select the View Applications tab, you will find a series of tabs displayed along the top of this pane. Clicking the tabs lets the administrator perform the following administrative actions on a selected application:

- Review the selected application's log file as it records events.
- Monitor the clients connecting to the application.
- View the streams and shared objects running in the application.
- Review the performance statistics for the computer where the application is running.
- Reload the application.

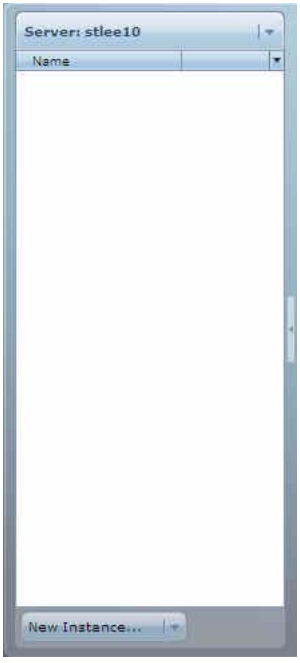
  The Reload button lets you reload an application instance that is currently connected to the server. You might do this to reload the instance's server-side scripts or to disconnect all of its users while immediately allowing new connections.
- Unload the application.

  The Unload button lets you drop an application instance. To unload an application instance, select it from the Applications menu and click Unload. This disconnects all clients to the instance.

  If the application has more than one instance running on the server, only the selected instance is stopped.

## Creating a new application instance

In the View Applications section of the management console, you can create a new application instance by selecting the New Instance button.

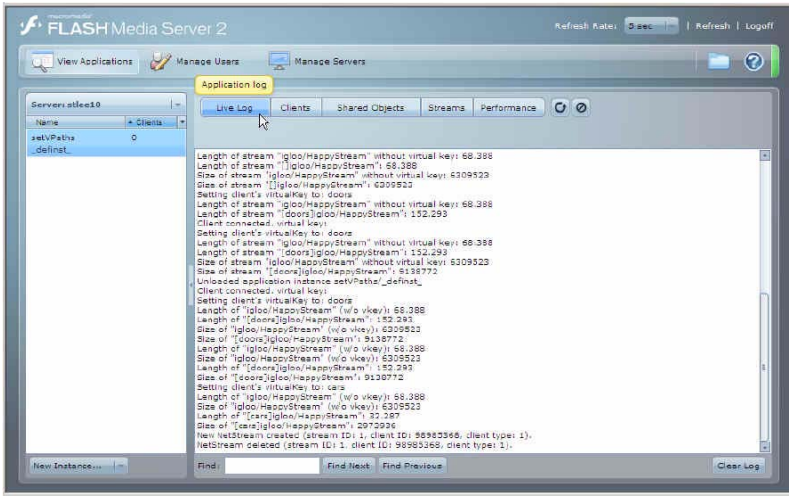



This action creates a new application instance within the application list. The management console adds a default instance suffix _definst_, which can be edited. Press Enter to submit the name and start the application instance. To cancel, press Escape+Shift.

This pane is persistent in every screen in the View Applications section of the management console. The pane can be resized and collapsed.

## Viewing the Live [Application] log file

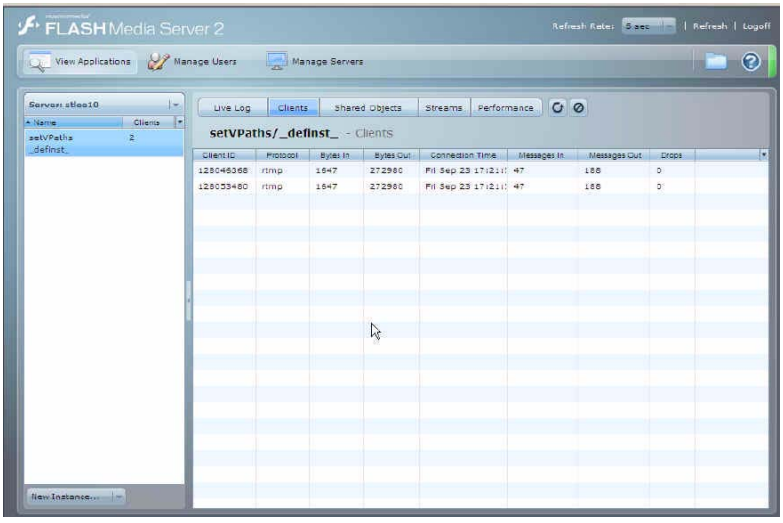
Each application creates an associated log file. The Live [Application] log pane displays the log messages.




The application administrator can use the Find box on the bottom margin of the pane to search for partial strings in the log messages. Pressing the Clear Log button clears the log view.

## Viewing active clients

This pane lists all client connections including debugging connections to the selected application.



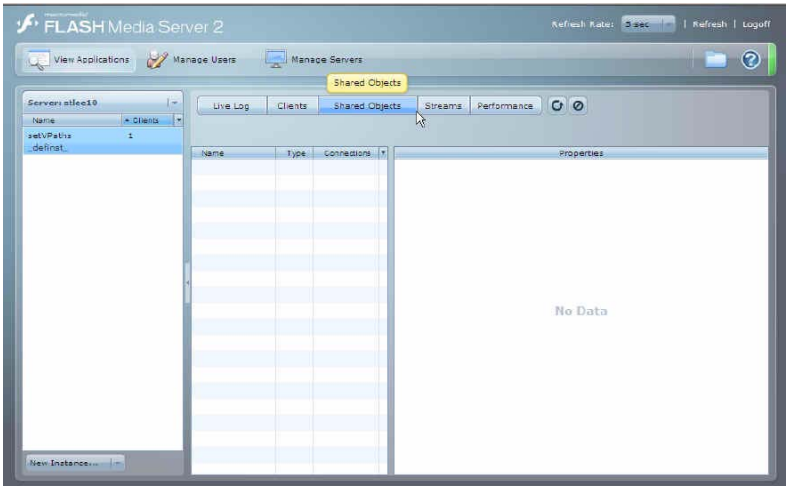

The management console displays the following information for each client:

- Client ID
- Connection protocol
- Number of bytes in the connection request and the information returned
- Connection time
- Number of messages in and out of the application
- Dropped messages

The management console displays the same information in the Manage Servers section.

## Viewing active shared objects

This pane lists the active shared objects for an application. The management console displays their name, type (persistent or volatile), and connections (number of users subscribed to this shared object). Select a shared object to view its data values.

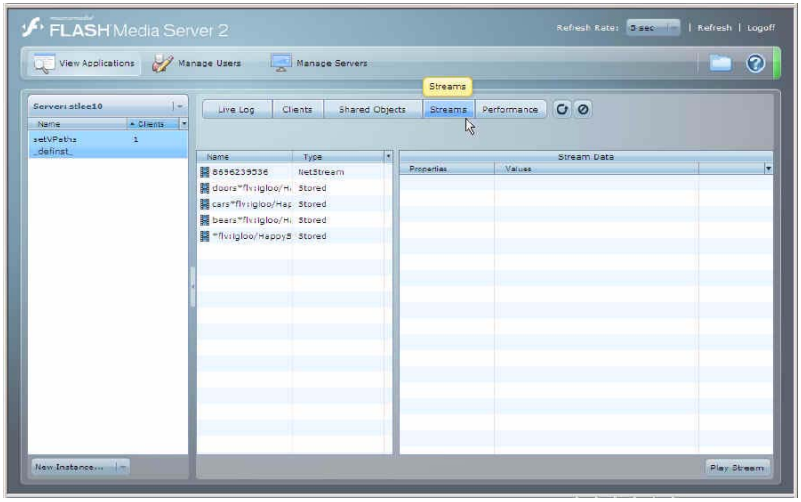



The information on this pane is helpful when debugging the application. When you select a shared value, one of the following occurs:

- If the shared object is available for debugging the application, the management console displays its properties in the adjoining pane.
- If the shared object is not available for debugging, a pop-up window appears stating it is not available for debugging.

## Viewing active streams

This pane lists all the active streams in the selected application. The management console displays their names and type. Select a stream to view its properties.




To play back a stream, select it and click the Play Stream button on the bottom margin of the pane. The Play Stream button appears if a debug connection is possible. If debugging is not allowed, the Play Stream button does not show.

The Viewing server performance tab pertains to the performance of this particular application.The information on this pane is helpful when debugging the application.

When you select a type, one of the following occurs:

- If the type is available for debugging the application, the management console displays its properties in the adjoining pane.
- If the type is not available for debugging, a pop-up window appears stating it is not available for debugging.

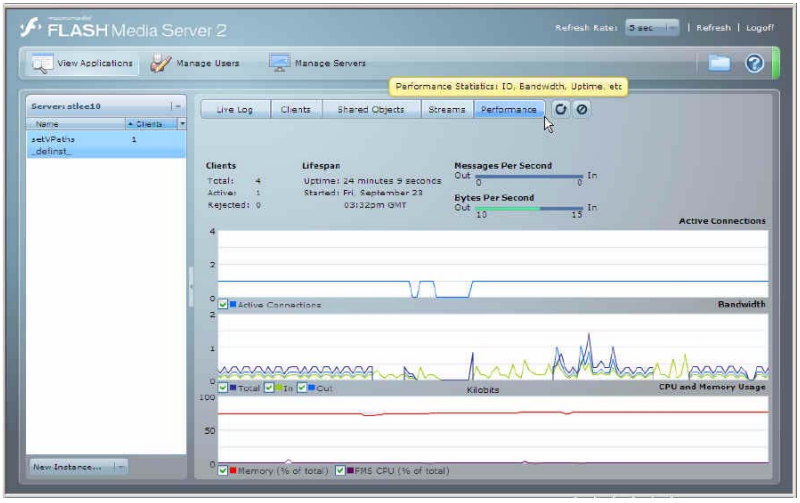
## Viewing application performance

This pane displays the live information for this application. Application and server administrators can review the following data:

- Client information: total number of clients, how many connections are active, how many connection requests were rejected.
- Life span of the application: indicating the time the application was started and how long it has been running continuously.
- Number of messages in and out of the application.
- Amount of bytes in and out of the application.

This pane also graphically displays the following information for Flash Media Server:

- Number of active connections it is supporting.
- Bandwidth resources it is consuming.
- Percentage of its allocated CPU and memory resources being consumed.



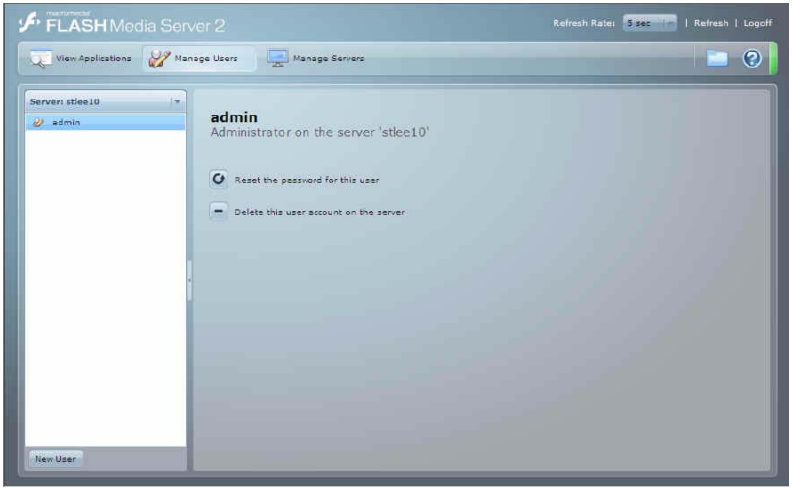

The information on this pane is helpful when debugging the application. The performance statistics are useful when allocating resources and optimizing performance.

# Managing the administrative users

In this section of the management console you control Flash Media Server users with administrative permissions. You can perform the following actions:

- Add new server and virtual host administrators.
- Delete administrators.
- Reset administrators' passwords.

The Users pane occupies the left side of the Manage Users section of the management console. The right pane provides detailed information when you select an administrator in the left pane.




This pane lists the accessible servers and virtual hosts.

- Select a server or virtual host to display its authorized administrators.
- The right-side pane lists the administrators for the server or virtual hosts.
- Select the New User button to add an administrator for the server
- Select an administrator on the Users panel Users panel to reset their password or delete them from the server or virtual host.
  - When you click the Reset the password for this user link, a prompt appears asking for the new password. You can now reset this user's password or Cancel the action.
  - When you click the Delete this user account on the server link, a prompt appears asking you to confirm the action. You can now delete this administrator's account on the server or virtual host or Cancel the action.

# Managing the servers

The Server pane occupies the left side of the Manage Server section of the management console. This pane lists the servers and virtual hosts that the administrator can access and manage.


This pane allows the administrator to select an individual server or a group of servers for viewing information. Servers are grouped into a tree structure. The sample shows the presence of the server (stlee10) and five virtual hosts. As the administrator enrolls additional servers or virtual hosts, they are listed here.

You will find a series of buttons displayed along the top of this pane:

Clicking one of the buttons lets the administrator perform one of the following administrative actions on a selected server:

- Add a server to the administrator's list.
- Edit the login information for a server.
- Delete a server on the administrator's list.

You will also find a series of buttons displayed along the base of this pane:

Clicking the buttons lets the administrator perform the following administrative actions on a selected server or virtual host:

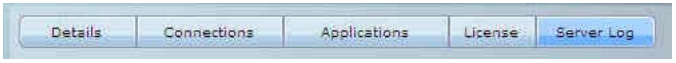
- Connect the management console to the selected server.
- The console can be connected to multiple servers simultaneously.
- Ping the selected server to verify that the server is running and view its responsiveness in milliseconds.
- Restart (or start) the selected server or virtual host.
- Check for and remove unused servers or virtual hosts.
- Stop the selected server.

Virtual host administrators can only shut down the applications on their own virtual host or restart that virtual host.

## Managing servers

This tab allows the administrator to review the performance of individual servers or a group of servers. The design of the panel permits the administrator to monitor a group of servers rapidly. The servers are grouped into a tree structure of individual servers.

When you select the Manage Servers tag, you will find a series of tabs displayed along the top of this pane:



Clicking the buttons lets the administrator perform the following actions:

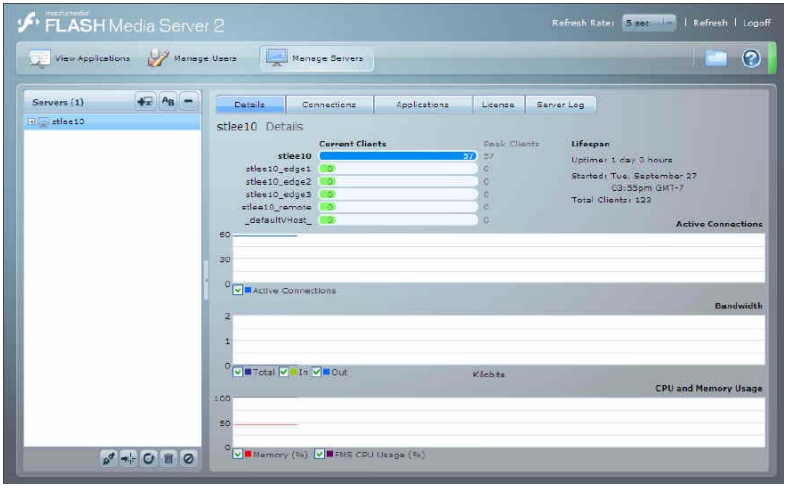
- Review the performance statistics for the computer where the applications are running.
- Review detailed information on the connections to the server.
- Review detailed information about the applications located on the server or virtual host.
- Review the server's license keys and files.
- Review the server's Access log file as it records connections.

The Server pane also occupies the left side of the screen in this section of the management console. The pane lists the servers and virtual hosts that the administrator can access and manage.

## Viewing server details

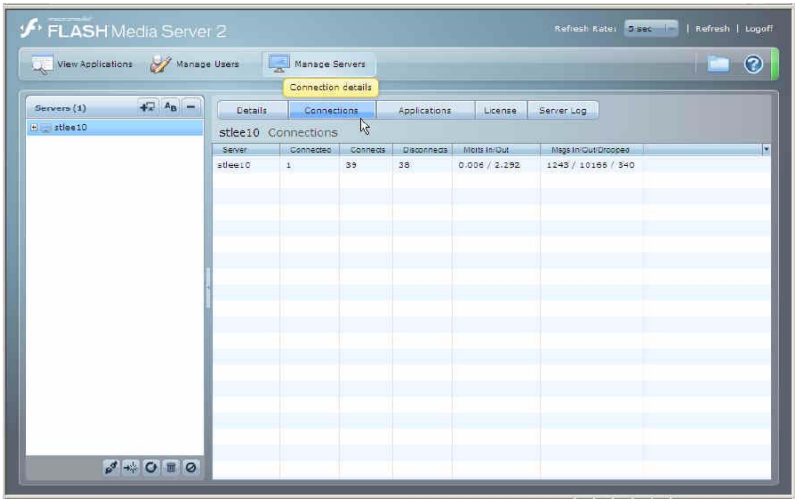
This pane displays live information for this application and the server. Administrators can review the following performance data:

- Client information: total number of clients, how many connections are active, how many connection requests were rejected.
- Life span of the server.
- Number of messages in and out of the server.
- Amount of bytes in and out of the server.
- Graphical displays of active connections, bandwidth resources consumed, and CPU and memory resources consumed.



## Viewing connection details

This pane lists all client connections to the selected server.



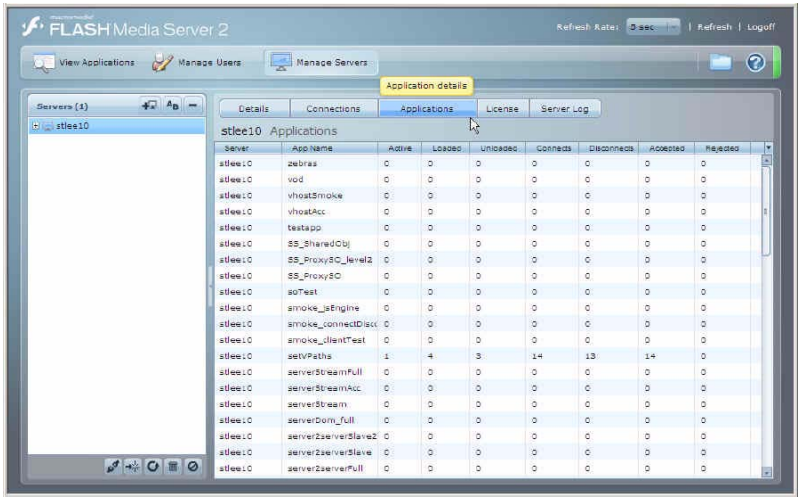

The management console displays the following information for each client accessing the server or virtual host:

- Client ID.
- Connection protocol.
- Number of bytes in the connection request and the information returned.
- Connection time.
- Number of messages in and out of the application.
- Number of messages dropped.

The management console also displays this information on a per-application basis in the Clients pane in the View Applications section.

## Viewing application details

This pane displays detailed information for all the applications running on the selected server or virtual host.

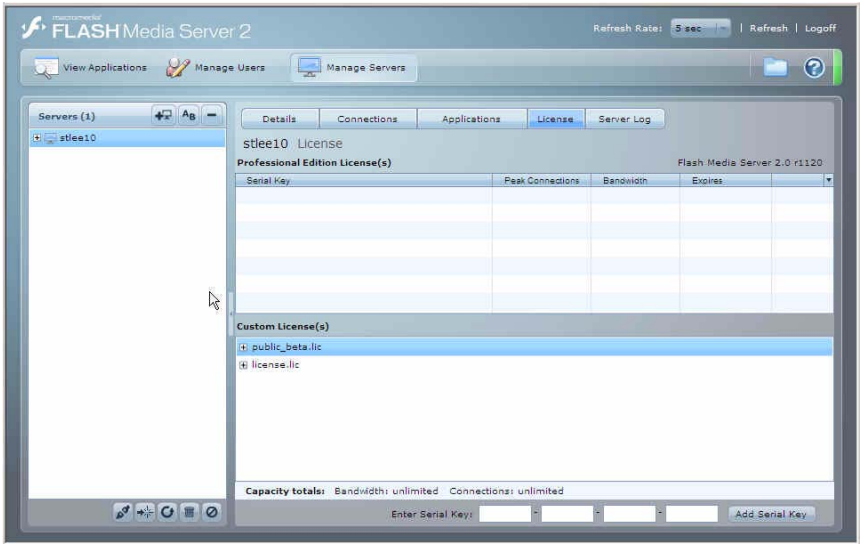



This panel displays current information about the application instances that are running on the server. The name of each application is displayed, along with the number of instances of the application that have been loaded on or unloaded from the server, the number of users that are connected, and the total number of connections that have been accepted and rejected for each application.

If you are connected as a virtual host administrator, Update displays information only for the virtual host you are connected to.

## Viewing license files

This pane displays detailed information for all license files authorizing you to run Flash Media Server on the selected server or virtual host.




On this panel the management console displays the detailed information for your Flash Media Server license. Select an individual license to display its details in the lower frame.

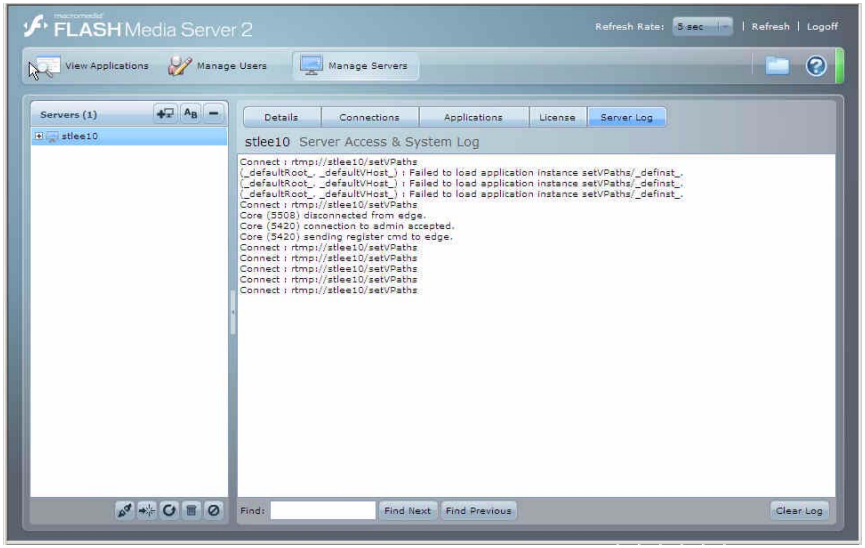
For each serial key, the management console displays the following information:

- Authorized peak number of client connections.
- Current bandwidth cap.
- License's expiration date, if applicable.

Your organization may have more than one license. Note the display of capacity totals.

## Viewing the server log file

This panel displays the trace messages being recorded in the server log file. The log messages report errors as well as normal operations.




The server administrator can use the Find box on the bottom margin of the pane to search for partial strings in the log messages. Clicking the Clear Log button clears the screen.

# Logging client connections and other system events

Flash Media Server 2 provides a logging functionality that allows the server to record information about client connections and other system events. The following sections describe the sections in the Logger.xml configuration files.

- "Access log file" on page 35
- "Application log file" on page 39
- "Diagnostic log file" on page 40
- "Configuring logging" on page 52

## Access log file

Flash Media Server 2 maintains an access log that includes statistics about client connections and stream activity. Flash Media Server 2 also maintains application logs for application activities and application logs for diagnostic logs. The application and diagnostic logs are an addition to operating system logs that log error and informational messages about Flash Media Server 2 operations.

The access log records information about requests by Flash Player and Flash Media Server 2 application instances. Using these logs, you can find out when a user connected to the server, the total bandwidth consumed during the session, and information about resources such as the streams accessed by the connection. You can use this information to determine which applications are the most frequently used.

The default access log is access.xx.log, which is located in the Flash Media Server 2 logs directory. The default configuration for Flash Media Server 2 creates a single access log per server. You can also configure Flash Media Server 2 to create a separate file per vhost. When logging is configured on a per-vhost basis, all logs for a particular vhost are found in a subdirectory within the logs directory. The name of the subdirectory matches the vhost name. Substitution strings can be found in the [] brackets, YYYY, MM, DD, and NN representing year, month, date, and version. You can use the substitution string to customize the filename of the access log.

To configure the server to create separate log files for each vhost, set the value of the `Scope` tag in the Server.xml file to "vhost" (see the description of the tag "Scope" on page 117).

For more information on logging, see "Logger.xml file" on page 135.

Flash Media Server 2 defines event categories, and for each category it defines events that can be recorded. Logging can be customized to record all events or only specific events. This determines how much information is collected.

The following table lists the access events defined in the Access logs.

| Event | Category | Description |
|---|---|---|
| connect-pending | application | Client connects to the server, waiting for the script to authenticate. |
| connect | application | Client connects to the server. |
| disconnect | application | Client disconnects. |
| publish | application | Client publishes a live stream. |
| unpublish | application | Client unpublishes a live stream. |
| play | application | Client plays a recorded or live stream. |

| Event | Category | Description |
|---|---|---|
| pause | application | Client pauses playing a stream. |
| unpause | application | Client resumes playing a stream. |
| seek | application | Client jumps to a new location within a recorded stream. |
| stop | application | Client stops playing a recorded or live stream or stops publishing a live stream. |
| record | application | Client begins the recording of a stream. |
| recordstop | application | Client stops the recording of a stream. |
| server-start | application | Server has started. |
| server-stop | application | Server has stopped. |
| vhost-start | application | A virtual host has started. |
| vhost-stop | application | A virtual host has stopped. |

The following table lists the fields in the access logs.

NOTE

When the data for this field contains a space or delimiter, the data is wrapped in double quotation marks. The double quotation marks surrounding the data are not part of the data but are present for better parsing of the data. This applies to the tz, x-ctx, x-adaptor, x-vhost, s-uri, c-referrer, c-user-agent, cs-bytes, sc-bytes, and x-sname fields.

| Field | Event(s) | Description |
|---|---|---|
| x-event | application | Type of event. |
| x-category | application | Event category. |
| date | application | Date at which the event occurred. |
| time | application | Time at which the event occurred. |
| tz | application | Time zone information. |
| x-ctx | application | Event-dependent context information. |
| x-pid | application | Server process ID. |
| x-cpu-load | application | CPU load. |
| x-mem-load | application | Memory usage (as reported by the `getServerStats()` method). |
| x-adaptor | application | Adaptor name. |
| x-vhost | application | Virtual host name. |
| x-app | application | Application names. |

| Field | Event(s) | Description |
|---|---|---|
| x-appinst | application | Application instance names. |
| c-ip | application | Client IP address. |
| c-proto | application | Connection protocol: RTMP or RTMPT. |
| s-uri | application | URI of the Flash Media Server 2 application. |
| c-referrer | application | URI of the referrer. |
| c-user-agent | application | User agent. |
| c-client-id | application | Client ID. |
| cs-bytes | application | This field shows the number of bytes transferred from the client to the server.<br>This information can be used to bill customers per session. To calculate the bandwidth usage per session, subtract the 'cs-bytes' in the 'connect' event from the 'cs-bytes' in the 'disconnect' event. |
| sc-bytes | application | This field shows the number of bytes transferred from the server to the client.<br>This information can be used to bill customers per session. To calculate the bandwidth usage per session, subtract the 'sc-bytes' in the 'connect' event by the 'sc-bytes' in the 'disconnect' event |
| x-sname | application | Stream name. |
| x-file-size | application | Stream size in bytes. |
| x-file-length | application | Stream length in seconds. |
| x-spos | application | Stream position. |
| cs-stream-bytes | application | This field shows the number of bytes transferred from the client to the server per stream.<br>To calculate the bandwidth usage per stream, subtract the 'cs-stream-bytes' in the 'publish' event by the 'cs-stream-bytes' in the 'unpublish' event. |
| sc-stream-bytes | application | This field shows the number of bytes transferred from the server to the client per stream.<br>To calculate the bandwidth usage per stream, subtract the 'sc-stream-bytes' in the 'play' event by the 'sc-stream-bytes' in the 'stop' event. |
| cs-uri-stem | application | Stem portion of s-uri (omitting query) field. |
| cs-uri-query | application | Query portion alone of s-uri. |

| Field | Event(s) | Description |
|---|---|---|
| x-sname-query | application | Query portion of stream URI specified in play or publish. |
| x-file-name | application | Full path of the file representing x-sname stream. |
| x-file-ext | application | Stream type (currently this can be flv or mp3). |
| s-ip | application | IP address or addresses of the server. |
| x-duration | application | Duration of a stream or session event. |
| x-suri-query | application | Same as x-sname-query. |
| x-suri-stem | application | This is a composite field: cs-uri-stem + x-sname + x-file-ext. |
| x-suri | application | This is a composite field: cs-uri-stem + x-sname + x-file-ext + x-sname-query. |
| x-status | application | For a complete description of the x-status codes and descriptions, see the following table. |

The following events display a status code.

| Field | Status Code | Description |
|---|---|---|
| connect-pending | 100 | Waiting for the application to authenticate. |
| connect | 200 | Successful connection. |
| | 302 | Application currently unavailable. |
| | 400 | Bad request; client connected to server using an unknown protocol. |
| | 401 | Connection rejected by the application script. |
| | 403 | Connection rejected by access module. |
| | 404 | Application not found. |
| | 409 | Resource limit exceeded. |
| | 413 | License limit exceeded. |
| | 500 | Server internal error. |
| | 502 | Bad gateway. |
| | 503 | Service unavailable; for instance, too many connections pending for authorization by access module. |

| Field | Status Code | Description |
|---|---|---|
| play | 200 | Successful. |
| | 400 | Bad request (invalid arguments). |
| | 401 | Access denied by application. |
| | 403 | Play forbidden by stream module. |
| | 404 | Stream not found. |
| | 415 | Unsupported media type. |
| | 500 | Server internal error. |
| publish | 200 | Successful. |
| | 400 | Bad request (invalid arguments). |
| | 401 | Access denied by application. |
| | 409 | Stream is already being published |
| | 415 | Unsupported media type. |
| | 500 | Server internal error. |
| stop | 200 | Successful. |
| | 408 | Stream stopped because client disconnected. |

## Application log file

The application log records information about activities in application instances. This log is used primarily for debugging (logging uncommon events that occur in an application instance). Users can also add user-defined messages to the application log by using trace() in server-side scripting.

The default application log is application.xx.log, located in the subdirectory within the Flash Media Server 2 logs directory. Flash Media Server 2 is configured, by default, to create one application log per application instance. The application folder is located in the matching vhost directory. The "xx" in the filename is a 2-digit number representing the history of the application log. The most recent logs can be found in "application.00.log."

The following table lists the fields in the application logs.

| Field | Event(s) | Description |
|---|---|---|
| date | all | Date on which the event occurred. |
| time | all | Time at which the event occurred. |

| Field | Event(s) | Description |
|---|---|---|
| x-pid | all | Server process ID. |
| x-status | all | Status code: the code is a 10-character string that represents the severity, category and message ID.<br><br>The first 3 characters represent severity. This is always in a letter format. The letters are as follows:<br><br>(w) = warning<br>(e) = error<br>(i) = information<br>(d) = debug<br>(s) = trace from server-side script<br>(_) = unknown<br><br>The next 3 characters represent category. All categories are listed in the "Status" table below for the diagnostic logs.<br><br>The last 4 characters represent message ID. All messages IDs are listed in the "Status Message IDs" table below for the Diagnostics logs. |
| x-ctx | all | Event-dependent context information |

## Diagnostic log file

The diagnostic log records alternative information (alternative to the information logged by the operating system) about Flash Media Server 2 operations. This log is used primarily for debugging (logging uncommon events that occur in Flash Media Server 2 processes). There are give different types of processes in Flash Media Server 2: master, edge, core, admin, and httpcache.

The default dialog logs are master.xx.log, edge.xx.log, core.xx.log, admin.xx.log, and httpcache.xx.log. All of the diagnostic logs are located in the Flash Media Server 2 logs directory. Flash Media Server 2 is configured, by default, to create a diagnostic log for each type of process. The "xx" in the file name is a 2-digit number presenting the version of the log. The most recent logs can be found in "version.00.log."

The following table lists the fields in the diagnostic logs.

| Field | Event(s) | Description |
|---|---|---|
| date | all | Date on which the event occurred. |
| time | all | Time at which event occurred. |
| x-pid | all | Server process ID. |
| x-status | all | Status code: the code is a 10-character string that represents the severity, category and message ID.<br><br>The first 3 characters represent severity. Always in a format of (letter). The letter can be any of the following:<br><br>(w) = warning<br>(e) = error<br>(i) = information<br>(d) = debug<br>(s) = trace from server-side script<br>(_) = unknown<br><br>The next 3 characters represent category. All categories are listed in the Status Category table below.<br><br>The last 4 characters represent message ID. All messages IDs are listed in the Status Message ID table below. |
| x-stx | all | Event-dependent context information |

The following table lists the status categories in the diagnostic logs.

| Category | Description |
|---|---|
| 257 | TCService |
| 258 | TCServer |
| 259 | Presence |
| 260 | Storage |
| 261 | Stream |
| 262 | SMTP |
| 263 | Adaptor |

| Category | Description |
|---|---|
| 264 | Javascript |
| 265 | TCApplication |
| 266 | TCConnector |
| 267 | Admin |
| 268 | SharedObject |
| 269 | Configuration |
| 270 | VirtualHost |
| 271 | SSL |

The following table lists the status message IDs in the diagnostic logs. Flash Media Server 2 uses the symbols %1$S, %2$S and %3$S as substitution strings in the status messages.

| Message ID | Description |
|---|---|
| 1000 | Received termination signal; server shutdown in progress. |
| 1001 | Received interrupt signal; server shutdown in progress. |
| 1002 | Server initialization failed; service will be stopped. |
| 1003 | Error during shutdown process; process will be terminated. |
| 1004 | Reinitializing server. |
| 1005 | Failed to start the following listeners for adaptor %1$S: %2$S |
| 1006 | Failed to stop %1$S listeners for adaptor %2$S. |
| 1007 | Failed to create thread (%1$S). |
| 1008 | Asynchronous I/O operation failed (%1$S: %2$S). |
| 1009 | Service Control Manager failed (%1$S: %2$S). |
| 1010 | Service Control Manager failed (%1$S: %2$S). |
| 1011 | Server starting... |
| 1012 | Server stopped %1$S. |
| 1013 | Failed to create listener for adaptor %1$S, IP %2$S, port %3$S: %4$S. |
| 1014 | Command name not found in the message. |
| 1015 | Method not found (%1$S). |
| 1016 | Failed to execute method (%1$S). |
| 1017 | Failed to stop virtual host (%1$S). |

| Message ID | Description |
|---|---|
| 1018 | The call method failed, invalid parameters: call(methodName[, resultObj, p1, pn]). |
| 1019 | Dropping application (%1$S) message. Clients not allowed to broadcast message. |
| 1020 | Response object not found (%1$S). |
| 1021 | Missing unlock for shared object %1$S, lock count %2$S. |
| 1022 | Nested lock for shared object %1$S, lock count %2$S. |
| 1023 | Unlock called without matching lock for shared object %1$S. |
| 1024 | Invalid application; rejecting message (%1$S). |
| 1025 | Ignoring message from client during authentication. |
| 1026 | Connection to %1$S lost. |
| 1027 | Unknown %1$S command issued for stream %2$S (application %3$S). |
| 1028 | Exception while processing message. |
| 1029 | Bad network data; terminating connection: %1$S |
| 1030 | Illegal subscriber: %1$S cannot subscribe to %2$S. |
| 1031 | Failed to start virtual host (%1$S). |
| 1032 | Failed to open configuration file: %1$S |
| 1033 | Parse error at line %1$S: %2$S. |
| 1034 | Connect failed ( %1$S, %2$S ): %3$S |
| 1035 | Invalid proxy object; connection may be lost (%1$S). |
| 1036 | Connect from host (%1$S) not allowed. |
| 1037 | No adaptors defined. |
| 1038 | Adaptor already defined with the name %1$S. |
| 1039 | Rejecting connection from %1$S to %2$S. |
| 1040 | Failed to create administrator: %1$S. |
| 1041 | Failed to remove administrator: %1$S. |
| 1042 | Failed to change password: %1$S. |
| 1043 | Resource limit violation. Unable to create stream: %1$S. |
| 1044 | Resource limit violation. Unable to create shared object: %1$S. |
| 1045 | Script execution is taking too long. |

| Message ID | Description |
|---|---|
| 1046 | Reserved property (%1$S). |
| 1047 | Admin request received from an invalid admin server. |
| 1048 | Administrator login failed for user %1$S. |
| 1049 | Failed to start server. |
| 1050 | Write access denied for shared object %1$S. |
| 1051 | Read access denied for shared object %1$S. |
| 1052 | Write access denied for stream %1$S. |
| 1053 | Read access denied for stream %1$S. |
| 1054 | Virtual host %1$S is not available. |
| 1055 | Invalid parameters to %1$S method. |
| 1056 | Alive |
| 1057 | NetConnection.Call.Failed |
| 1058 | Invalid application name (%1$S). |
| 1059 | Invalid user ID (%1$S). |
| 1060 | NetConnection.Admin.CommandFailed |
| 1061 | Invalid parameters to %1$S method. |
| 1062 | Failed to unload application %1$S. |
| 1063 | Failed to load application %1$S. |
| 1064 | %1$S applications unloaded. |
| 1065 | Admin user requires valid user name and password. |
| 1066 | Invalid virtual host alias : %1$S |
| 1067 | Error registering class: name mismatch (%1$S, %2$S). |
| 1068 | Connection rejected: maximum user limit reached for application instance %1$S. |
| 1069 | (%2$S, %3$S) : Failed to load application instance %1$S. |
| 1070 | (%2$S, %3$S) : Connection rejected to application instance %1$S: client already connected to an application. |
| 1071 | Illegal access property (%1$S). |
| 1072 | %1$S is now published. |
| 1073 | %1$S is now unpublished. |

| Message ID | Description |
|---|---|
| 1074 | Stopped recording %1$S. |
| 1075 | Stream %1$S has been idling for %2$S second(s). |
| 1076 | Playing and resetting %1$S. |
| 1077 | Pausing %1$S. |
| 1078 | Unpausing %1$S. |
| 1079 | Started playing %1$S. |
| 1080 | Stopped playing %1$S. |
| 1081 | Recording %1$S. |
| 1082 | Failed to record %1$S. |
| 1083 | New NetStream created (stream ID: %1$S). |
| 1084 | NetStream deleted (stream ID: %1$S). |
| 1085 | Publishing %1$S. |
| 1086 | Failed to publish %1$S. |
| 1087 | Failed to restart virtual host (%1$S). |
| 1088 | Connection to Flash Media Server 2 has been disconnected. |
| 1089 | Failed to play (stream ID: %1$S). |
| 1090 | Failed to play %1$S (stream ID: %2$S). |
| 1091 | Play stop failed, stream ID: %1$S. |
| 1092 | Audio receiving enabled (stream ID: %1$S). |
| 1093 | Audio receiving disabled (stream ID: %1$S). |
| 1094 | Failed to enable audio receiving (stream ID: %1$S). |
| 1095 | Failed to stop playing (stream ID: %1$S). |
| 1096 | Video receiving enabled (stream ID: %1$S). |
| 1097 | Video receiving disabled (stream ID: %1$S). |
| 1098 | Set video fps to %1$S (stream ID: %2$S). |
| 1099 | Failed to receive video (stream ID: %1$S). |
| 1100 | Seeking %1$S (stream ID: %2$S). |
| 1101 | Failed to seek (stream ID: %1$S). |
| 1102 | Failed to seek %1$S (stream ID: %2$S). |
| 1103 | Invalid schedule event format (%1$S). |

| Message ID | Description |
|---|---|
| 1104 | Invalid method name (%1$S). |
| 1105 | (%2$S, %3$S): Invalid application name (%1$S). |
| 1106 | Connection succeeded. |
| 1107 | Connection failed. |
| 1108 | Invalid shared object (%1$S). |
| 1109 | Unknown exception caught in %1$S. |
| 1110 | Invalid stream name (%1$S). |
| 1111 | Server started (%1$S). |
| 1112 | JavaScript runtime is out of memory; server shutting down instance (Adaptor: %1$S, VHost: %2$S, App: %3$S). Check the JavaScript runtime size for this application in the configuration file. |
| 1113 | JavaScript engine runtime is low on free memory. Take action. |
| 1114 | Failed to start listeners for adaptor %1$S. |
| 1115 | Configuration error for adaptor %1$S: IP %2$S and port %3$S are already in use. |
| 1116 | Failed to create adaptor: %1$S. |
| 1117 | Failed to play %1$S; stream not found. |
| 1118 | Insufficient admin privileges to perform %1$S command. |
| 1119 | Failed to initialize listeners for adaptor %1$. Flash Media Server 2 is already running or other processes are using the same ports. |
| 1120 | Configuration file not found: %1$S |
| 1121 | Invalid configuration file: %1$S |
| 1122 | Server aborted. |
| 1123 | Invalid NetStream ID (%1$S). |
| 1124 | Failed to open shared object file (%1$S) for write. |
| 1125 | Failed to open shared object file (%1$S) for read. |
| 1126 | Failed to flush shared object (%1$S). |
| 1127 | Failed to initialize shared object from persistent store (%1$S). |
| 1128 | Invalid shared object file (%1$S). |
| 1129 | Failed to play %1$S; index file not found or mismatch. |
| 1130 | (%2$S, %3$S): Application (%1$S) is not defined. |

| Message ID | Description |
|---|---|
| 1131 | (%2$S, %3$S): Resource limit violation. Unable to load new application: %1$S. |
| 1132 | (%2$S, %3$S): Resource limit violation. Unable to create new application instance: %1$S. |
| 1133 | (%2$S, %3$S): Resource limit violation. Rejecting connection to: %1$S. |
| 1134 | Failed to load admin application. |
| 1135 | Preload application aborted. |
| 1136 | (%2$S, %3$S): Application (%1$S) is currently offline. |
| 1137 | Admin command setApplicationState failed for %1$S. |
| 1138 | Command Successful. |
| 1139 | Script is taking too long to process the event. Shutting down instance: %1$S. |
| 1140 | NetConnection.Call.Success |
| 1141 | Unable to locate server configuration file during startup. |
| 1142 | Unable to locate script file: %1$S. |
| 1143 | NetConnection.Call.AccessDenied |
| 1144 | NetConnection.Call.BadValue |
| 1145 | Publish %1$S failed, invalid arguments. |
| 1146 | Pause %1$S failed, invalid arguments. |
| 1147 | Unable to create directory %1$S. |
| 1148 | Server shutdown failed. |
| 1149 | Invalid admin command: %1$S. |
| 1150 | Beta expired. |
| 1151 | Invalid object name (stream ID: %1$S). |
| 1152 | Breaking potential deadlock, shared object(%1$S) lock reset to unlocked. |
| 1153 | Potential deadlock, shared object (%1$S) has been locked for %2$S sec. |
| 1154 | Invalid license key: %1$S |
| 1155 | License key specified does not allow multiple adaptor support. |
| 1156 | License key specified does not allow multiple virtual host support. |

| Message ID | Description |
|---|---|
| 1157 | (%2$S, %3$S/%1$S): Current server bandwidth usage exceeds license limit set. Rejecting connection. |
| 1158 | (%2$S, %3$S/%1$S): Current virtual host bandwidth usage exceeds max limit set. Rejecting connection. |
| 1159 | Multiprocessor support available only in enterprise edition. |
| 1160 | Trial run expires Server shutting down. |
| 1161 | License key has expired. |
| 1162 | Invalid shared object name (%1$S). |
| 1163 | Failed to record %1$S, no space left on device |
| 1164 | Unknown exception occurred. Instance will be unloaded: %1$S |
| 1165 | Failed login attempt from %1$S at %2$S. |
| 1166 | Attempt to reconnect to Flash Media Server 2. |
| 1167 | Failed to remove application: %1$S. |
| 1168 | Exception while processing message: %1$S |
| 1169 | Failed to execute admin command: %1$S |
| 1170 | Unloaded application instance %1$S |
| 1171 | System memory load (%1$S) is high. |
| 1172 | System memory load (%1$S) is now below the maximum threshold. |
| 1173 | %1$S |
| 1174 | Listener started (%1$S): %2$S |
| 1175 | Restarting listener (%1$S): %2$S |
| 1176 | Out of memory: %1$S. |
| 1177 | Adaptor (%1$S) has an SSL configuration error on port %2$S. |
| 1178 | Error from %1$S:%2$S. |
| 1179 | Warning from %1$S:%2$S. |
| 1180 | Info from %1$S:%2$S. |
| 1181 | Exception caught in %1$S while processing streaming data inside %2$S. |
| 1182 | (%2$S, %3$S): Max connections allowed exceeds license limit. Rejecting connection to: %1$S. |
| 1183 | An internal version control error has occurred. |

| Message ID | Description |
|---|---|
| 1184 | Invalid cryptographic accelerator: %1$S. |
| 1185 | Failed to initialize cryptographic accelerator: %1$S. |
| 1186 | Failed to seed the pseudo-random number generator. |
| 1187 | Application directory does not exist: %1$S |
| 1188 | Using default application directory: %1$S |
| 1189 | Application instance is not loaded: %1$S |
| 1190 | Error: command message sent before client connection has been accepted. |
| 1191 | Failed to play %1$S; adaptor not found: %2$S. |
| 1192 | Invalid value set for configuration key: %1$S = %2$S, using %3$S. |
| 1193 | Pending queue size limit %1$S reached. Rejecting connection request Host: %2$S:%3$S. |
| 1194 | Client to server bandwidth limit exceeded. [Virtual host (%1$S), Max Allowed %2$S, Current %3$S] |
| 1195 | Server to client bandwidth limit exceeded. [Virtual host (%1$S), Max Allowed %2$S, Current %3$S] |
| 1196 | Adaptor (%1$S) does not exist. |
| 1197 | Virtual host (%1$S) does not exist. |
| 1198 | Message queue is too large. Server memory usage too high. Disconnecting client. |
| 1199 | Duplicate license key: %1$S |
| 1200 | Expired license key: %1$S |
| 1201 | No primary license key found. Switching to Developer Edition. |
| 1202 | Commercial and Educational licenses can not be mixed. Switching to Developer Edition. |
| 1203 | Personal and Professional licenses can not be mixed. Switching to Developer Edition. |
| 1204 | NFR licences can not be mixed with any other kind. Switching to Developer Edition. |
| 1205 | OEM licences can not be mixed with any other kind. Switching to Developer Edition. |
| 1206 | Too many trial licenses detected. Switching to Developer Edition. |

| Message ID | Description |
|---|---|
| 1207 | Shared object %1$S has changed and is not being saved as auto commit is set to false. Current version %2$S, Last saved version %3$S. |
| 1208 | %1$S failed. Invalid argument %2$S. |
| 1209 | File operation %1$S failed. %2$S |
| 1210 | File operation %1$S failed. File is in closed state (%2$S). |
| 1211 | File operation %1$S failed. Object is not a file (%2$S). |
| 1212 | File object creation failed (%1$S). |
| 1213 | Connection rejected by server. Reason: %1$S. |
| 1214 | Invalid substitution variable: %1$S |
| 1215 | Resetting service failure action from %1$S to %2$S. |
| 1216 | Administrator (%1$S) already exists. |
| 1217 | Failed to open log file, log aborted. |
| 1218 | Failed to play stream %1$S: Recorded mode not supported. |
| 1219 | Missing arguments to %1$S method. |
| 1220 | Invalid admin stream: %1$S |
| 1221 | Core (%1$S) started, arguments: %2$S. |
| 1222 | Failed to start core: %1$S %2$S |
| 1223 | Core (%1$S) is no longer active. |
| 1224 | Edge (%1$S) started, arguments: %2$S. |
| 1225 | Failed to start edge: %1$S %2$S |
| 1226 | Edge (%1$S) is no longer active. |
| 1227 | Shared memory heap (%1$S) has exceeded 90 usage. Consider increasing the heap size to prevent future memory allocation failures. |
| 1228 | Failed to create process mutex. |
| 1229 | Process (%1$S): shared memory (%2$S) init failed. |
| 1230 | Process (%1$S): failed to map view of shared memory (%2$S). |
| 1231 | Core (%1$S) connected to admin. |
| 1132 | Core (%1$S) failed to connect to admin. |
| 1233 | Core (%1$S) disconnecting from admin. |
| 1234 | Core (%1$S) connection to admin accepted. |

| Message ID | Description |
|---|---|
| 1235 | Core (%1$S) connection to admin failed. |
| 1236 | Core (%1$S) received close command from admin. |
| 1237 | Starting admin app on core (%1$S). |
| 1238 | Core (%1$S) connecting to admin. |
| 1239 | Core (%1$S): Failed to initiate connection to admin. |
| 1240 | Core (%1$S) shutdown failed. |
| 1241 | Connection to admin received. |
| 1242 | Core (%1$S) disconnected: %2$S. |
| 1243 | Connection from core %1$S received. |
| 1244 | Connection from core %1$S accepted. |
| 1245 | Failed to send connect response to core %1$S. |
| 1246 | Core (%1$S) sending register command to edge. |
| 1247 | Core (%1$S) disconnected from edge. |
| 1248 | Core (%1$S) failed to establish proxy to edge. |
| 1249 | Core (%1$S) socket migration failed. |
| 1250 | Edge disconnected from core (%1$S). |
| 1251 | Proxy to core (%1$S) failed. |
| 1252 | Registering core (%1$S). |
| 1253 | Socket migration to core (%1$S) failed. |
| 1254 | Recovering edge %1$S with %2$S failure[s] after %3$S seconds! |
| 1255 | Edge (%1$S) %2$S experienced %3$S failure[s]! |
| 1256 | Core (%1$S) %2$S experienced %3$S failure[s]! |
| 1257 | Core (%1$S) %2$S is not responding and is being restarted! |
| 1258 | Core (%1$S) is no longer active; create a new one. |
| 1259 | Recovering core %1$S with %2$S failure[s] after %3$S seconds! |
| 1260 | Core (%1$S) did not shut down as expected. Killing core now. |
| 1261 | Command (%1$S) timed out. |
| 1262 | OpenProcess(PROCESS_TERMINATE) failed with %1$S. |
| 1263 | OpenProcess(PROCESS_QUERY_INFORMATION) failed for pid (%1$S) with %2$S |

# Configuring logging

Flash Media Server logging is configured through the Server.xml and Logger.xml configuration files. Server.xml contains a `Logging` section that controls the overall logging behavior. This section includes an `Enable` tag that determines whether logging takes place, and a `Scope` tag that determines whether Flash Media Server writes separate log files for each virtual host or one file for the entire server.

The following is an excerpt of the Logging section of the Server.xml file:

```
<Root>
  <Server>
    <Logging>
    <!-- Overall logging configuration. This section contains only -->
    <!-- information that controls the overall logging behavior. -->
    <!-- Specific logging configuration is located in Logger.xml. -->
      <!-- The time field was added after the Flash Comm Server 1.5.2 -->
      <!-- release. The server can log file in utc (gmt) or local time. -->
      <!-- The default time is local time. -->
      <Time>local</Time>
      <Access>
        <!-- Whether access logging is enabled. -->
        <Enable>true</Enable>
        <!-- The logging scope determines whether a log file is written -->
        <!-- out for each vhost or just one for the entire server. -->
        <!-- It may be either server or vhost. -->
        <Scope>server</Scope>
      </Access>
      <Diagnostic>
        <!-- Whether diagnostic logging is enabled. -->
        <Enable>true</Enable>
      </Diagnostic>
      <Application>
        <!-- Whether application logging is enabled. -->
        <Enable>false</Enable>
      </Application>
    </Logging>
  </Server>
<Root>
```

Logger.xml files may be provided at the configuration root folder right next to Server.xml and optionally for each vhost right next to VHost.xml file. The root Logger.xml file determines the logger configuration when the logging scope is server-wide. Optionally a specific vhost Logger.xml controls the logging behavior for a given vhost. The virtual host-specific Logger.xml configuration file is relevant only when the activities for each virtual host are being logged in a separate log file. The location of each log file is determined by the `Directory` and `FileName` tags in the Logger.xml file(s).

A sample of the Logger.xml file is included later in this document.

For a more complete listing of all tags, see “Logger.xml file” on page 135.

## Logger.xml file example

The Logger.xml file contains the following XML:

```
<Logger>
  <Access>
    <!-- Directory in which log files will be placed. By default -->
    <!-- they are placed in logs/ in the server installation directory. -->
    <Directory>${LOGGER.LOGDIR}</Directory>

    <!-- Access log file name, everything inside the square brackets [] >
    <!-- will be substituted -->
    <!-- Y represents Year, only YYYY is allowed -->
    <!-- M represents Month, only M or MM are allowed -->
    <!-- D represents Day, only D or DD are allowed -->
    <!-- N represents Version, there is no limit on number of N's -->
    <!-- The number of each letter represents number of digits, for -->
    <!-- example, April in M is 4 and in MM is 04 -->
    <FileName>access.[YYYYMMDDNN].log</FileName>

    <!-- The time field in a log file can be either in utc or local -->
    <!-- The setting here can be used to override the server-wide
    <!-- configuration, See <Logging> in Server.xml. -->
    <!-- <Time></Time> -->

    <Rotation>
      <!-- Maximum file size in kilobytes (1024 bytes)-->
      <MaxSize>10240</MaxSize>

      <!-- Rotation Time, there are 2 types-->
      <!-- If Type="daily", rotation only occurs every 24 hours, -->
      <!-- and the format is hh:mm, for example 00:00 will rotate -->
      <!-- every midnight -->
      <!-- If type="duration", rotation occurs when the duration of -->
      <!-- the log exceeds a certain length. -->
      <!-- Duration takes a number in minutes. -->
      <Schedule type="daily">00:00</Schedule>

      <!-- Max number of log files to keep. Files will be named -->
      <!-- access.log.1, access.log.2, access.log.3 etc. -->
      <History>5</History>
    </Rotation>

    <!-- The following describes which events that can be logged. -->
    <!-- The various events are as follows: -->
    <!-- Event Name                      Category-->
    <!-- ==========                      ========-->
    <!--  1. connect-pending              session -->
```

```
   <!--  2. connect                      session -->
   <!--  3. disconnect                   session -->
   <!--  4. publish                      stream -->
   <!--  5. unpublish                    stream -->
   <!--  6. play                         stream -->
   <!--  7. pause                        stream -->
   <!--  8. unpause                      stream -->
   <!--  9. seek                         stream -->
   <!-- 10. stop                         stream -->
   <!-- 11. server-start                 server -->
   <!-- 12. server-stop                  server -->
   <!-- 13. vhost-start                  vhost -->
   <!-- 14. vhost-stop                   vhost -->
   <!-- The desired events are specified as a semi-colon separated list --
>
   <!-- Specifying * will log all events. -->

   <Events>connect;disconnect;play;pause;unpause;stop</Events>


   <!-- The following describes which information gets logged for each  --
>
   <!-- event. Not all fields make sense for all events in which case   --
>
   <!-- they will be empty in the log file. The possible fields are -->
   <!--    1. x-event          Type of event                          -->
   <!--    2. x-category       Event category                         -->
   <!--    3. date             Date at which the event occurred       -->
   <!--    4. time             Time at which the event occurred       -->
   <!--    5. tz               Time zone information            -->
   <!--    6. x-ctx            Event-dependent context information    -->
   <!--    7. x-pid            Server process id                       -->
   <!--    8. x-cpu-load       CPU load                               -->
   <!--    9. x-mem-load       Memory load (as reported in
getServerStats)-->
   <!--   10. x-adaptor        Adaptor name                           -->
   <!--   11. x-vhost          Vhost name                             -->
   <!--   12. x-app            Application name                       -->
   <!--   13. x-appinst        Application instance name            -->
   <!--   14. c-ip             Client ip address                      -->
   <!--   15. c-proto          Connection protocol - rtmp or rtmpt    -->
   <!--   16. s-uri            URI of the fms application             -->
   <!--   17. c-referrer       URI of the referrer                    -->
   <!--   18. c-user-agent     User agent                             -->
   <!--   19. c-client-id      Client ID                              -->
   <!--   20. cs-bytes         Bytes transferred from client to server ->
   <!--   21. sc-bytes         Bytes transferred from server to client ->
   <!--   22. x-sname          Stream name                            -->
   <!--   23. x-file-size      Stream size in bytes                   -->
   <!--   24. x-file-length    Stream length in seconds               -->
```

```
    <!--   25. x-spos           Stream position                          -->
    <!--   26. cs-stream-bytes  Stream bytes transferred client to server->
    <!--   27. sc-stream-bytesStream bytes transferred server to client-->
    <!--   28. s-ip             IP address(es) of the server            -->
    <!--   29. x-duration       Duration of an event/session            -->
    <!--   30. x-status         Status an event                         -->
    <!--   31. cs-uri-stem      Stem of an s-uri                        -->
    <!--   32. cs-uri-query     Query portion of s-uri                  -->
    <!--   33. x-sname-query    Query portion of stream uri             -->
    <!--   34. x-file-name      Full file path of recorded stream       -->
    <!--   35. x-file-ext       Stream file type (flv or mp3)           -->
    <!--   36. x-suri-query     Same as x-sname-query-->
    <!--   37. x-suri-stem      cs-uri-stem + x-sname + x-file-ext -->
    <!--   38. x-suri           x-suri-stem+ x-suri-query-->
<!-- The field specification is a semicolon-separated list of one  -->
    <!-- or more field names. The special keyword * indicates that all -->
    <!-- fields are to be logged. When customizing the fields to be     -->
    <!-- logged, it is strongly recommended to always at least log      -->
    <!-- the type, category, date, and time fields.                      -
  ->

    <Fields>x-category;x-event;date;time;c-ip;cs-bytes;sc-bytes;x-
  sname;sc-stream-bytes;x-file-size;x-file-length</Fields>

    <!-- Delimiter is used to separate between fields                    --
  >
    <!-- Recommended: tab or ' '.                                        --
  >
    <!-- If no delimiter is specified, default is tab                    --
  >
    <!-- Not Recommended: '#', because it is used as comment             --
  >
    <!-- tag in W3C format                                               --
  >
    <!-- Disallowed: ''', '"', '.', ':', '-', '\' and all alpha-numeric  --
  >
    <!-- (a-z, A-Z, 0-9).  The characters ':' and '-' are not allowed    --
  >
    <!-- because ':' is used in the time field and '-' is used in the    -->
    <!-- date field and in fields with empty value.                      --
  >
    <Delimiter>','</Delimiter>

    <!-- This is an optional flag to control if the fields need to be    --
  >
    <!-- quoted when space or the delimiter are found in the fields.     --
  >
    <!-- It can be set to enable or disable.  By default, it is set to   -->
    <!-- disable.                                                        -
  ->
```

```
    <QuoteFields>disable</QuoteFields>
    <!-- This is an optional flag to control if the fields need to be -->
    <!-- escaped when unsafe characters are found in the fields.  It can --
  >
    <!-- be set to enable or disable. By default, it is set to enable. -->
    <!-- Based on rfc1738, unsafe characters are space, <, >, ", #, %,
  {,},-->
    <!--|, -^, ~, [, ], ` -->
    <EscapeFields>enable</EscapeFields>

  </Access>

<Diagnostic>
    <!-- Directory in which log files will be placed, by default they are -
  ->
    <!-- placed in logs/ in the server installation directory. -->
    <Directory>${LOGGER.LOGDIR}</Directory>

    <Rotation>
      <!-- Maximum file size in kilobytes (1024 bytes) -->
      <MaxSize>10240</MaxSize>

      <!-- Rotation Time, there are 2 types -->
      <!-- If type="daily", rotation only occurs every 24 hours, and the -->
      <!-- format is hh:mm, for example 00:00 will rotate every midnight -->
      <!-- If type="duration", rotation occurs when the duration of the log
  -->
      <!-- exceeds a certain length, duration takes a number in minutes -->
      <Schedule type="daily">00:00</Schedule>

      <!-- Max number of log files to keep, files will be named
  admin.01.log, -->
      <!-- admin.02.log, admin.03.log etc. -->
      <History>5</History>
    </Rotation>
  </Diagnostic>
  <Application>
    <!-- Directory in which log files will be placed, by default they are -
  ->
    <!-- placed in logs/ in the server installation directory. -->
    <Directory>${LOGGER.LOGDIR}</Directory>

    <Rotation>
      <!-- Maximum file size in kilobytes (1024 bytes) -->
      <MaxSize>10240</MaxSize>

      <!-- Rotation Time, there are 2 types -->
      <!-- If type="daily", rotation only occurs every 24 hours, and the -->
      <!-- format is hh:mm, for example 00:00 will rotates every midnight -
  ->
```

```
      <!-- If type="duration", rotation occurs when the duration of the log
  -->
      <!-- exceed a certain length, duration takes a number in minutes -->
      <Schedule type="daily">00:00</Schedule>

      <!-- Max number of log files to keep, files will be named -->
      <!-- application.01.log, application.02.log etc. -->
      <History>5</History>
    </Rotation>
  </Application>
</Logger>
```

NOTE

Log file rotation cannot be disabled. To effectively turn off rotation, however, you can choose a large maximum size and a long maximum duration.

## Viewing server events in the Windows event viewer

You can also use the Windows event viewer for tracking Flash Media Server activity and debugging server applications. The event viewer displays a list of events that the server generates.

### To use the Windows event viewer:

1. From the Windows Start menu, select Settings > Control Panel > Administrative Tools > Event Viewer.
2. Select the Application panel.
3. Double-click an event generated by Flash Media Server to view the details of the event.

## Configuring the server at runtime

Using the Server Management ActionScript API, you can view and edit the server's configuration settings by building your own customized administration applications. You can add or remove administrators, change their user names and passwords, and change most of the other server settings in all four of the server's XML files. For detailed information about using these ActionScript commands, see the Flash Support Center at www.macromedia.com/support/flash/.

# Managing Flash Media Server on Linux

On all supported Linux platforms, Flash Media Server 2 is installed as a service and includes a command-line utility, the fmsmgr utility, to perform certain administration tasks.

You must be a root user to install the server and manage it using the fmsmgr utility; for more information, see "Using the fmsmgr utility" on page 58.

The default ports for the server and the Admin service are, respectively, 1935 and 1111. The default option is to run the server as non-root user 'nobody.' If you don't want to run the server as user 'nobody,' choose a user with a valid account on the system. To specify which user to run the server as after installation, edit the `Process` tags in the Server.xml configuration file. Be sure the user you specify has permission to read/write the server files. For more information, see "Server.xml file" on page 86.

## Starting the Flash Media Admin Service in Windows

The Flash Media Admin Service is the service that you connect to when you log on to the server through the management console. It is installed automatically when you install the server.

The Admin service must be running to allow users to access the management console. You must be a root user to start the Admin service.

To start the Admin service, see "Using the fmsmgr utility" on page 58.

## Starting the Flash Media Admin Service on Linux

The Flash Media Admin Service is configured by default to start when the host system is started. You can change this setting using the `fmsmgr` utility.

You cannot configure an Admin service to start automatically; the Admin service must be started manually by a root user.

## Using the fmsmgr utility

Use the `fmsmgr` utility to perform basic management tasks for the Flash Admin Service running on Linux systems. You must be a root user to use the `fmsmgr` utility.

### Syntax

```
fmsmgr server <service_name> <cmd>
```

The following table describes the commands for the `fmsmgr` utility.

| Command | Description |
|---|---|
| `fmsmgr server adminserver start\|stop\|abort\|restart` | Starts, stops, restarts, or aborts the Flash Admin Service. |
| `fmsmgr server clearautostart` | Sets the Flash Admin Service to be started manually.<br>This command affects only the server service; Admin services cannot be started automatically. |
| `fmsmgr server fms getadmin` | Gets the name of the Flash Admin Service and indicates whether or not that service is running. |
| `fmsmgr list` | Lists all the services installed, including Admin Services, with additional information about services that are currently running. |
| `fmsmgr fms server remove`<br>`fmsmgr fms adminserver remove` | Removes the Flash Media Server 2 service or the Admin service from the fmsmgr tables. If you remove a server service, the corresponding Admin service is also removed.<br>Warning: Use this command only if you want to uninstall the server; you still need to manually remove the installed files. |
| `fmsmgr server [service_name] abort` | Stops a running Flash Media Server service. *`service_name`* is the name of the server you selected during installation. Processes, such as streaming or garbage collection, are not allowed to complete.<br>If no name is specified, the action is performed on the default server. |
| `fmsmgr server [`*`service_name`*`] restart` | Stops a running Flash Media Server service and restarts it. If no name is specified, the action is performed on the default server. If the default *service_name* doesn't exist, the command fails. |
| `fmsmgr server [`*`service_name`*`] stop` | Stops the specified Flash Media Server service. *`service_name`* is the name of the server you selected during installation. If no name is specified, the action is performed on the default server. If the default *service_name* doesn't exist, the command fails. |
| `fmsmgr server [`*`service_name`*`] start` | Starts the Flash Media Server service. *`service_name`* is the name of the server you selected during installation. |

| Command | Description |
| --- | --- |
| `fmsmgr setadmin service_name` | Changes the default Admin service. *service_name* is the name of the server you selected during installation. The Admin service name is the same as the Flash Media Server 2 service name. Any installed Admin service can be used to administer one or more servers. Only one Admin service can be running at a time. |
| `fmsmgr setautostart` | Sets the Flash Media Server service to start automatically when the system is started. |

For other administrative tasks, such as adding users or checking the status of applications, you use the management console. For more information, see "Using the management console" on page 16. Although you do not need to be a root user to use the management console, the Admin service itself does need to be started by a root user using the fmsmgr utility before anyone can use the management console.

CHAPTER 2

# Deploying Flash Media Server

2

This chapter describes the various strategies for deploying Macromedia Flash Media Server 2, including the use of edge and origin servers.

Flash Media Server has been designed to accommodate many types of media applications. After installation, the server's configuration files contain only simple, generic settings. You'll need to make some decisions about how to configure the server to best suit your organization's requirements.

## Typical configurations

Flash Media Server can be used in a variety of different ways, with different configurations. In most cases, the server is used with a web server. Applications that run on Flash Media Server consist of clients developed in Macromedia Flash—that is, SWF files. Each application defined on the server has a corresponding directory that contains the streams and scripts used by the application.

Your web server is responsible for serving the SWF client files and the HTML pages in which they are embedded. In addition, you may use an application server along with your web server and Flash Media Server to incorporate database or other features into your media applications.

If you are running Flash Media Server on a Linux system, you must also have a Windows or Macintosh computer with Flash to create Flash Media Server applications. Your client-side files (SWF and HTML files) for your applications can reside on the Flash Media Server computer or another computer, depending on where your web server is installed. Your source files and server-side scripts (ASC, FLA, FSO, and FLV files) must reside on the Linux computer where Flash Media Server is running.

## Configuration for development and testing

While developing and testing your applications, you may choose to install a web server, Flash Media Server, and Flash on the same computer. The web root directory in this scenario would contain all the Flash Media Server elements of your applications, such as its FLA, SWF, HTML, script, stream, and shared object files. This configuration provides a simple working environment for designing and testing applications, and is possible only when you install Flash Media Server on a Windows computer.

For security reasons, this configuration is not recommended for deployment. In deployment, your FLA, script, stream, and shared object files would not be stored in the web root directory.

## Deploying on one computer

A relatively simple deployment scenario consists of one computer where a web server, Flash Media Server, and a firewall are installed. The firewall provides security for the server computer and the rest of your local network. In this and any other deployment scenario, the server-side files (ASC files), the audio/video files (Flash Video or FLV files), and the source files (FLA files) should not reside in the web server's published directories. These files should be located in your registered application directory within the applications directory in the Flash Media Server 2 directory (or, if you changed this location, the directory specified in the `AppsDir` tag in the Vhost.xml file). The web server's web root directory should contain only the HTML and SWF files for your applications.

If you are developing applications for Linux systems, you should use a two-computer deployment. For more information, see "Deploying on two computers" on page 62.

## Deploying on two computers

In this scenario, Flash Media Server and the application server are installed on two separate computers. This separation of files and functions provides more processor bandwidth. The web server computer can also function as an application server if your situation requires one.

If you are running Flash Media Server on a Linux system, you should use this deployment for development and testing, in which case you would also install Flash for Windows or Macintosh on the web server computer.

## Deploying on two computers with authentication through Flash Media Server

Some scenarios may require authentication of users who want to access information on an application server. In this case you may want to use a separate computer for Flash Media Server, and another for the web server and application server. Your Flash Media Server can perform the authentication and then retrieve data from the web/application server. This scenario requires the developer to create server-side scripts to perform these functions.

## Deploying on two computers with authentication through an application server

You may decide to have users authenticated before they are allowed to connect to your Flash Media Server. In this scenario, users connect first to the application server. After they are authenticated, the application server creates a ticket that the user uses to connect to Flash Media Server. The application must be designed to check for these tickets, typically with server-side scripts.

# SSL support in Flash Media Server

Secure Sockets Layer (SSL) is a protocol for enabling secure communications over TCP/IP. Flash Media Server provides native support for both incoming and outgoing SSL connections. Flash Media Server specifies the connection as secure by using the RTMPS protocol in the connect URL. To generate the required certificates, Flash Media Server uses a third-party open source library such as OpenSSL.

Since the size of Flash Player files must be kept to a minimum, embedding an SSL library in the player is not an option. Instead, Flash Media Server takes advantage of platform libraries such as WinINET on Windows to perform HTTP tunneling. An RTMPS connection from Flash Player to Flash Media Server is actually a secure HTTP connection (HTTPS).

## Defining a secure port

You can configure a port as secure by specifying a minus sign before the port number in the `HostPort` tag in the Adaptor.xml file.

```
<HostPort>:1935,80,-443</HostPort>
```

This XML tag specifies that Flash Media Server will listen on any interface on ports 1935, 80, and 443, where 443 is designated as a secure port that will receive only RTMPS connections. An RTMPS connection attempt to ports 1935 or 80 will fail: the client will attempt to perform an SSL handshake that the server will fail to complete. Similarly, a regular RTMP connection to port 443 will fail because the server will try to perform an SSL handshake that the client will fail to complete.

For more information, see "HostPort" on page 152.

## Configuring SSL

To configure Flash Media Server for secure communications, you use the `SSL` tags in the Server.xml and Adaptor.xml files.

The following tags in the `SSL` section of the Server.xml file configure Flash Media Server to act as an SSL-enabled client, making outgoing connections secure.

```
<SSL>
  <SSLEngine></SSLEngine>
  <SSLRandomSeed></SSLRandomSeed>
  <SSLSessionCacheGC></SSLSessionCacheGC>
    <SSLClientCtx>
      <SSLVerifyCertificate></SSLVerifyCertificate>
      <SSLCACertificatePath></SSLCACertificatePath>
      <SSLCACertifcateFile></SSLCACertificateFile>
      <SSLVerifyDepth></SSLVerifyDepth>
      <SSLCipherSuite><SSLCipherSuite>
    </SSLClientCtx>
</SSL>
```

For detailed information on configuring Server.xml for SSL, see "SSL" on page 119.

The `SSL` tags in the Adaptor.xml file configure Flash Media Server to act as an SSL-enabled server to accept incoming SSL-enabled connections.

```
<SSL>
  <SSLClientCtx>
    <SSLVerifyCertificate></SSLVerifyCertificate>
    <SSLCACertificatePath></SSLCACertificatePath>
    <SSLCACertifcateFile></SSLCACertificateFile>
    <SSLVerifyDepth></SSLVerifyDepth>
    <SSLCipherSuite><SSLCipherSuite>
  </SSLClientCtx>
</SSL>
```

For detailed information on configuring Adaptor.xml for SSL, see "SSL" on page 160.

## Creating multiple certificates for an adaptor

You can configure Flash Media Server to return multiple certificates on a given adaptor by configuring a certificate for each edge server:

- Configure each `HostPort` tag in the Adaptor.xml file with a `name` attribute.
- Configure each `HostPort` tag to return its own certificate by specifying an `Edge` tag under the `SSL` tag with a `name` attribute.
- Match the value for this `name` attribute to the `name` attribute of the `HostPort` tag for this certificate.

For example, suppose you had the following two `HostPort` tags:

```
<HostPort name="edge1" ctl_channel=":19350">:1935,-443</HostPort>
<HostPort name="edge2" ctl_channel=":19351">:1936,-444</HostPort>
```

In this case, the `SSL` tag would contain the following information:

```
<SSL>
  <SSLServerCtx>
  <SSLCertificateFile>cert.pem</SSLCertificateFile>
  <SSLCertificateKeyFile>private.pem</SSLCertificateKeyFile>
  <SSLPassPhrase></SSLPassPhrase>
  <SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH</SSLCipherSuite>
  <SSLSessionTimeout>5</SSLSessionTimeout>
  </SSLServerCtx>
  <Edge name="edge1">
    <SSLServerCtx>
      <SSLCertificateFile>cert2.pem</SSLCertificateFile>
      <SSLCertificateKeyFile>private2.pem</SSLCertificateKeyFile>
      <SSLPassPhrase></SSLPassPhrase>
      <SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH<SSLCipherSuite>
      <SSLSessionTimeout>5</SSLSessionTimeout>
    </SSLServerCtx>
  </Edge>
</SSL>
```

This sample code demonstrates how to configure `"edge1"` to return `cert2.pem` when a client connects to it on port 443. Since there is no `Edge` tag for `"edge2"`, `"edge2"` will use the default configuration specified in the `SSLServerCtx` section that is directly under the `SSL` container tag. The `"edge2"` server returns `cert1.pem` when a client connects to it on port 444.

## Configuring independent virtual hosts for SSL application

You can configure the different virtual hosts in Flash Media Server to manage its remote SSL connections independently. For example, you can disable certificate checking in one virtual host, use a different certificate in another store for its trusted root Certificate Authority (CA) certificates, and implement a different set of ciphers in a third virtual host.

To implement these independent settings, you need to specify the `SSL` section under the `Proxy` tag in the appropriate Vhost.xml file. When the `SSL` tag is present, the entire `SSL` section is used to configure the virtual host. If an `SSL` tag is omitted from this section, Flash Media Server uses the default settings.

```
<SSL>
  SSLClientCtx>
    <SSLVerifyCertificate></SSLVerifyCertificate>
    <SSLCACertificatePath></SSLCACertificatePath>
    <SSLCACertifcateFile></SSLCACertificateFile>
    <SSLVerifyDepth></SSLVerifyDepth>
    <SSLCipherSuite><SSLCipherSuite>
  </SSLClientCtx>
</SSL>
```

# About configuration levels

The server is capable of hosting more than one adaptor and more than one virtual host on each adaptor. Each virtual host is equivalent to a domain name. Each virtual host can run more than one media application. Some editions of the server are limited to one adaptor and one virtual host.

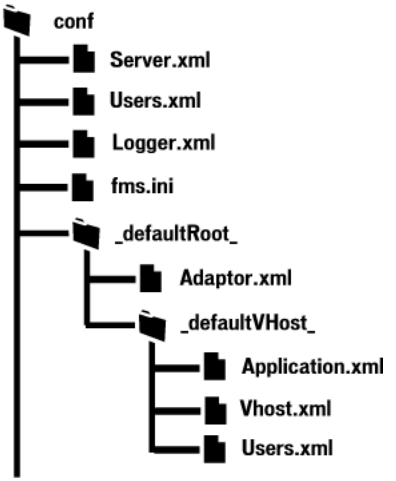
## About the configuration hierarchy

Flash Media Server can support several adaptors, virtual hosts, and applications simultaneously. Each adaptor on the server can serve multiple virtual hosts, and each virtual host can host multiple applications. By supporting multiple adaptors and virtual hosts, Flash Media Server facilitates management of multiple websites that may have different configurations and administrators.

Each of these layers of service has its own configuration settings, stored in separate XML files. These files are stored in a directory structure that reflects the hierarchy of adaptors, virtual hosts, and applications you want to use with the server.

For detailed information on the configuration files, see Chapter 3, “Configuration Files.”

The default directory structure installed with the server looks like this:

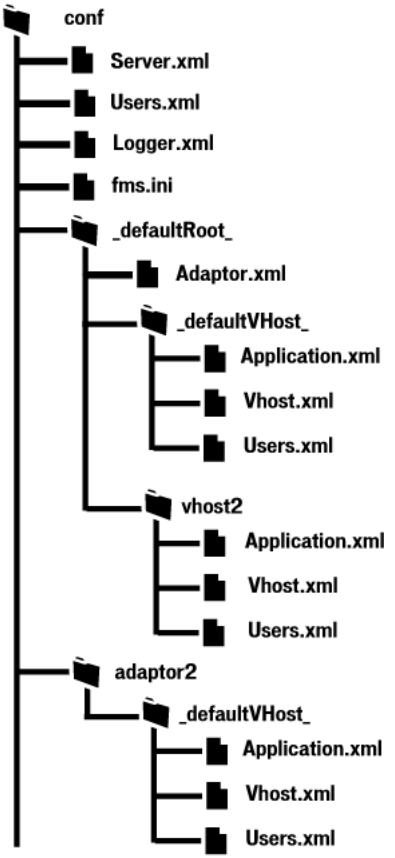



The directory structure includes three subdirectories: conf, _defaultRoot_, and _defaultVHost_.

- The `conf` subdirectory, at the top of the hierarchy, holds the configuration files for the server and the fms.ini file. This subdirectory contains the following:
  - The Server.xml file

    This file contains settings that relate to the server only. The specific settings for the adaptors, virtual hosts, and applications are stored in separate XML files.
  - The Users.xml file
  - The Logger.xml file
  - The fms.ini file
  - The _defaultRoot_ subdirectory
- The _defaultRoot_ subdirectory is the default adaptor directory for the server. It contains the Adaptor.xml file and the _defaultVHost_ subdirectory. The Adaptor.xml file contains the settings that relate to the adaptor.

If there is a second adaptor, it has its own subdirectory at the same level as the _defaultRoot_ subdirectory.

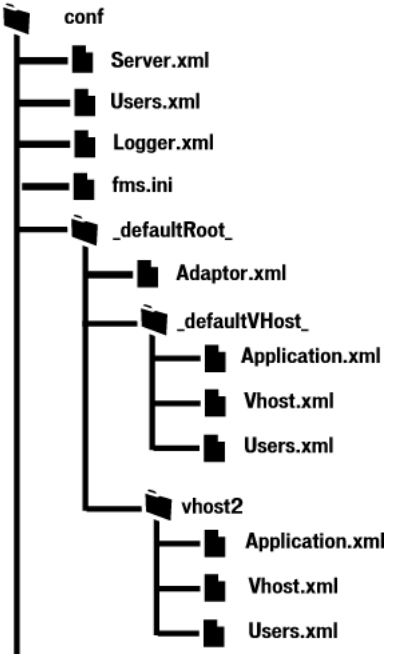



- The _defaultVHost_ subdirectory is the default virtual host subdirectory for the adaptor. It contains the Application.xml file, which contains default settings for the client applications that will connect to the server; the Vhost.xml file, which contains the settings for the virtual host; and the Users.xml file, which defines the administrative users and their permissions for the Flash Media Admin Service.

  The Users.xml file is required only if you are defining administrators for this virtual host.

  If there is another virtual host on the same adaptor, it has its own subdirectory at the same level as the _defaultVHost_ subdirectory.

Each adaptor directory must contain a _defaultVHost_ directory.




## Adding adaptors and virtual hosts

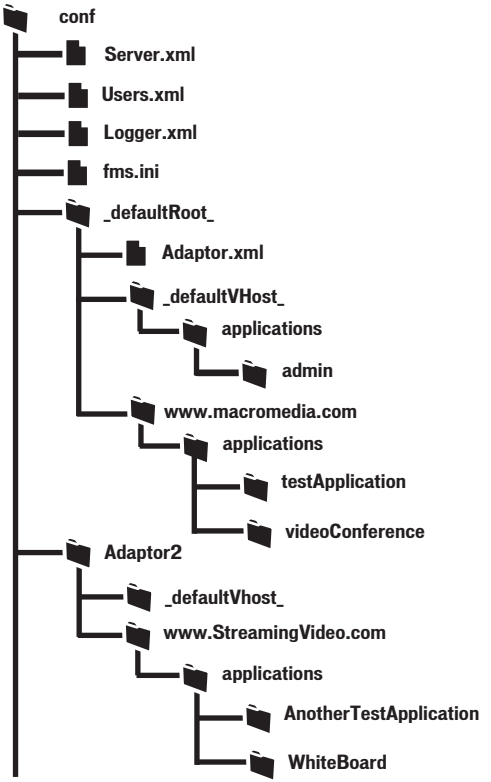
To add an adaptor to the server, you must add a complete adaptor directory structure to the server's conf directory. Each adaptor directory must contain an Adaptor.xml file and at least one virtual host directory, called _defaultVHost_. Any virtual hosts must be in addition to _defaultVHost_.

When you design an application in Flash that will connect to Flash Media Server, you add a subdirectory with the application's name to the application directory of the virtual host it will connect to. The application's subdirectory can contain its own Application.xml file if you want the application to override any of the settings in the generic Application.xml file in the virtual host directory.

To create a new virtual host, create a new virtual host directory inside the /conf/*adaptor_name* directory in the Flash Media Server directory, for the adaptor you want to use for the new virtual host: /conf/*adaptor_name*/*virtual_host_name*. This directory must include the following items:

- A Vhost.xml file
- An Application.xml file
- A Users.xml file, if you are defining administrators for this virtual host

A typical customized server conf directory might look like this:




*A customized conf directory containing multiple adaptors and application subdirectories and configuration files*

The conf directory illustrated here contains two adaptor subdirectories: the _defaultRoot_ subdirectory and the Adaptor2 subdirectory.

- The _defaultRoot_ subdirectory contains the Adaptor.xml file and the _defaultVHost_ subdirectory and another virtual host subdirectory named www.macromedia.com. Each of these virtual host subdirectories contains an application subdirectory. The application subdirectory for www.macromedia.com contains directories for the applications testApplication and videoConference.
- The Adaptor2 subdirectory contains its own _defaultVHost_ directory and another virtual host directory named www.streamingVideo.com. The www.streamingVideo.com subdirectory contains an application subdirectory with directories for AnotherTestApplication and a WhiteBoard application.

| TIP | The /conf/*adaptor_name*/*vhost_name*/applications directory needs to be specified in the fms.ini or Vhost.xml file. By default, the Flash Media Server 2 installer creates an applications directory under the installation directory and sets fms.ini to point here.<br><br>The /conf/*adaptor_name*/*vhost_name*/applications directory is different from the applications directory where you register your applications. |
|---|---|

# Server administration over HTTP

Flash Media Server 2 lets you administer the server over HTTP as well as Real-Time Messaging Protocol (RTMP). You can use the same server management application programming interface (API) over HTTP as you would over RTMP. By passing command strings and arguments to the URL of your Flash Media Server, you can interact with the server to retrieve information or modify the server configuration. This API is described in detail in the *Server Management ActionScript Language Reference*, included with Flash Media Server.

## Configuring Flash Media Server

A server administrator can control and configure the server by RTMP or HTTP or with the `User` and `AdminServer` tags in the Users.xml file.

To construct a URL that sends a command to the server, you must include the following:

- The directory *admin* after the server address and administration port number
- The administrator user name and password after the command

For example, the following URL passes a `ping` command to the server:

```
http://myFlashMediaServer:1111/admin/ping?auser=somename&apswd=somepassword
```

The server sends the results back to the browser in XML format.

```
<?xml version="1.0" encoding="utf-8" ?>
<result>
   <level>error</level>
   <code>Admin.Server.Disconnect</code>
   <timestamp>10/22/2003 05:31:01 PM</timestamp>
   <description>FMS server down.</description>
</result>
```

For example, the following URL sends a `getVhostStats` command to the server:

```
http://myFlashMediaServer:1111/admin/
  getVhostStats?auser=somename&apswd=somepassword&vhost="_defaultVhost_"
```

The following is the XML result:

```
<?xml version="1.0" encoding="utf-8" ?>
<result>
  <level>status</level>
  <code>NetConnection.Call.Success</code>
  <timestamp>11/17/2003 2:52:29 PM</timestamp>
  <data>
    <bytes_in>0</bytes_in>
    <bytes_out>3284</bytes_out>
    <bw_in>0</bw_in>
    <bw_out>0</bw_out>
    <msg_in>0</msg_in>
    <msg_out>1</msg_out>
    <msg_dropped>0</msg_dropped>
    <total_connects>1</total_connects>
    <total_disconnects>0</total_disconnects>
    <connected>1</connected>
    <accepted>1</accepted>
    <rejected>0</rejected>
    <total_apps>1</total_apps>
    <total_instances_loaded>1</total_instances_loaded>
    <total_instances_unloaded>0</total_instances_unloaded>
    <tunnel_bytes_in>0</tunnel_bytes_in>
    <tunnel_bytes_out>0</tunnel_bytes_out>
    <tunnel_requests>0</tunnel_requests>
    <tunnel_responses>0</tunnel_responses>
  </data>
</result>
```

For another example, the following URL sends a `getMsgCacheStats` command to the server:

```
http://myFlashMediaServer:1111/admin/
  getMsgCacheStats?auser=somename&apswd=somepassword&vhost="_defaultVhost_
  "
```

The following is the XML result:

```
<?xml version="1.0" encoding="utf-8" ?>
<result>
  <level>status</level>
  <code>NetConnection.Call.Success</code>
  <timestamp>10/22/2003 05:31:01 PM</timestamp>
  <data>
    <allocated>8588</allocated>
    <reused>6603</reused>
    <size>641100</size>
    <thread_count>10</thread_count>
    <units>
      <global_size>1434</global_size>
      <thread_size>703</thread_size>
      <size>2137</size>
      <reused>6603</reused>
      <allocated>8588</allocated>
      <released>2</released>
      <reallocated>6158</reallocated>
      <bulk_allocated>69</bulk_allocated>
      <bulk_released>1503</bulk_released>
      <huge_allocated>3</huge_allocated>
      <huge_released>2</huge_released>
    </units>
    <bytes>
      <global_size>430200</global_size>
      <thread_size>210900</thread_size>
      <size>641100</size>
      <reused>1980900</reused>
      <allocated>14068504</allocated>
      <released>1618633</released>
      <reallocated>11422687</reallocated>
      <bulk_allocated>20700</bulk_allocated>
      <bulk_released>450900</bulk_released>
      <huge_allocated>70317</huge_allocated>
      <huge_released>42185</huge_released>
    </bytes>
  </data>
</result>
```

| Field name | Description |
|---|---|
| `allocated` | Total number of messages allocated from the heap. |
| `bulk_allocated` | Total number of messages allocated from the global pool. |
| `bulk_released` | Total number of messages released back to the global pool. |
| `bytes` | Indicates that the following statistics are measured in number of bytes. |
| `global_size` | Size of the global pool free list. |

| Field name | Description |
|---|---|
| huge_allocated | Total number of huge (greater than 16K) messages allocated. |
| huge_released | Total number of huge messages released, in bytes. |
| reallocated | Total number of messages that have been reallocated. |
| released | Total number of messages released back to the heap. |
| reused | Total number of messages reused. |
| size | Total number of messages in the global and per-thread pool free lists. |
| thread_count | Total number of per-thread pools in use. |
| thread_size | Total size of all the per-thread pool free lists. |
| units | Indicates that the following statistics are measured in number of messages. |

# Using the admin commands

Many server administration commands expect one or more parameters. These are passed to the function as URL-encoded arguments. Arguments must adhere to the following formatting rules:

- Strings are passed as literals surrounded by quotation marks.You can use either single quotation marks (') or double quotation marks (").

  ```
  "Hello World"
  'String2'
  ```

  The only exceptions are the auser and apswd parameters, which should be passed as strings without quotation marks.
- Numbers are passed as either integer or floating-point literals.

  245

  1.03

  -45
- Arrays are passed as comma-separated values enclosed by square brackets.

  ```
  [1,2,3,4]
  ['abcd',34,"hi"]
  [10,20,[31,32],40]
  ```
- Objects are passed as JavaScript inline object literals.

  ```
  {foo:123,bar:456}
  {user:"Joe",ssn:"123-45-6789"}
  ```

When you use the server management API over HTTP, the arguments for each command must be named. This is different from ActionScript, where the arguments are not named because the meaning of each argument is determined by the order in which it is passed to the command.

For instance, the syntax for the `addAdmin` command is as follows:

```
:/admin/
  addAdmin?auser=adminname&apswd=adminpassword&username="joe"&password="
axbycz"&vhost="_defaultRoot_/foo.myCompany.com"
```

The syntax for the `gc` command is as follows:

```
/admin/gc?auser=adminname&apswd=adminpassword
```

The following table lists the names of the arguments for each command. Remember that the admin user name and admin password are required for every command.

| Command | Required arguments | Optional arguments | Sample URL |
|---|---|---|---|
| addAdmin | username, password | scope | /admin/addAdmin?username="joe"&password="axbycz"&vhost="_defaultRoot_/foo.myCompany.com" |
| addApp | appName | n/a | /admin/addApp?appName="app1" |
| addVHostAlias | vhost/vhostName, alias/aliasName, persist | n/a | /admin/addVHostAlias?vhost="_defaultVHost_"&alias="www.somealias.com"&persist=false |
| approveDebugSession | appInst, debugPIN | n/a | /admin/approveDebugSession?appName="app1/inst1"&debugPIN="1234" |
| broadcastMsg | scope, method | arg0, arg1, arg2 ... | /admin/broadcastMsg?scope="App:myApp"&method="myMethod"&arg0="foo"&arg1="bar" |
| changePswd | username, password | scope | /admin/broadcastMsg?username="admin1"&password="nimda1"&scope="_defaultRoot_/foo.macromedia.com" |

| Command | Required arguments | Optional arguments | Sample URL |
|---|---|---|---|
| disconnectUsers | appInst, clients | n/a | /admin/disconnectUsers?appInst="simpsons/game1"&clients=["13794136", "13799720"] |
| gc | n/a | n/a | /admin/gc |
| getActiveInstances | n/a | n/a | /admin/getActiveInstances |
| getAdaptors | n/a | n/a | /admin/getAdaptors |
| getAdminContext | n/a | n/a | /admin/getAdminContext |
| getApps | n/a | n/a | /admin/getApps |
| getAppStats | appName | n/a | /admin/getAppStats?appName="foo" |
| getConfig2 | key | scope | /admin/getConfig2?key="Admin/Server/UserList/User:scott/Password"&scope="/" |
| getFileCacheStats | n/a | n/a | /admin/getFileCacheStats |
| getGroupMembers | appInst, uid/userid | n/a | /admin/getGroupmembers?appInst="foo"&groupid=63741000 |
| getGroups | appinst | n/a | /admin/getGroups?appInst="foo" |
| getGroupStats | appInst, uid/userid | n/a | /admin/getGroupStats?appInst="so_test/_definst_"&groupid=63741000 |
| getInstanceStats | appInst | n/a | /admin/getInstanceStats?appInst="foo" |
| getIOStats | n/a | n/a | /admin/getIOStats |
| getLicenseInfo | n/a | n/a | /admin/getLicenseInfo |
| getLiveStreams | appInst | n/a | /admin/getLiveStreams?appInst="foo" |
| getLiveStreamStats | appInst, stream | n/a | /admin/getLiveStreamStats?appInst="foo"&stream="live1" |
| getMsgCacheStats | n/a | n/a | /admin/getMsgCacheStats |

| Command | Required arguments | Optional arguments | Sample URL |
|---|---|---|---|
| getNetStreams | appInst | n/a | /admin/ getNetStreams?appInst="foo" |
| getNetStreamStats | appInst, streamids | n/a | /admin/ getInstanceStats?appInst="recorderApp/ _definst_"&streamids=[1,2] |
| getRecordedStreams | appInst | n/a | /admin/ getRecordedStreams?appInst="foo" |
| getRecordedStreamsStats | appInst, stream | n/a | /admin/ getRecordedStreamStats?appInst="foo"&stream="on2key?flv:rec1" |
| getScriptStats | appInst | n/a | /admin/ getScriptStats?appInst="foo" |
| getServerStats | n/a | n/a | /admin/getServerStats |
| getServices | n/a | n/a | /admin/getServices |
| getSharedObjects | appInst | n/a | /admin/getUsers?appInst="foo" |
| getSharedObjectStats | appInst, sharedObject persistent | n/a | /admin/ getInstanceStats?appInst="so_test/ _definst_"&sharedObject="data_db"&persistent=false |
| getUsers | appInst | n/a | /admin/getUsers?appInst="foo" |
| getUserStats | appInst, userid | n/a | /admin/ getInstanceStats?appInst="so_test/ _definst_"&userid=63741000 |
| getVHosts | n/a | adaptor | /admin/ getVHosts?adaptor="_defaultRoot_" |
| getVHostStats | n/a | adaptor | /admin/ getVHostStats?adaptor="_defaultRoot_"&vhost="_defaultVHost" |
| ping | n/a | n/a | /admin/ping |

| Command | Required arguments | Optional arguments | Sample URL |
|---|---|---|---|
| reloadApp | appInst | n/a | /admin/reloadApp?appInst="foo" |
| removeAdmin | userName | scope | /admin/removeAdmin?userName="foo"&scope="server" |
| removeApp | appName | n/a | /admin/removeApp?appName="foo" |
| removeVHostAlias | vhost/vhostName, alias/aliasName, persist | n/a | /admin/removeVHostAlias?vhost="_defaultVHost_"&alias="www.somealias.com" |
| restartVHost | n/a | scope | /admin/restartVHost?scope="_defaultRoot_/foo.macromedia.com" |
| setConfig2 | key, value | scope | /admin/setConfig2?key="Admin/Server/UserList/User:scott/Password"&value="foo"&scope="/" |
| startServer | n/a | mode | /admin/startServer |
| startVHost | vhost | n/a | /admin/startVHost?vhost="_defaultRoot_/foo.macromedia.com" |
| stopServer | n/a | mode | /admin/stopServer?mode="normal" |
| stopVHost | n/a | vhost | /admin/stopVHost?vhost="_defaultRoot_/foo.macromedia.com" |
| unloadApp | appInst | n/a | /admin/unloadApp?appInst="foo" |

# Symbolic text substitutions

Flash Media Server supports the use of symbolic text substitutions in all tags in the server's XML configuration files. When you specify a symbol in any configuration tag, Flash Media Server will substitute the string you have mapped to that symbol when it reads the configuration file. After you have configured the server the first time, future edits will be easier: you can keep all the strings and symbol mappings in a single file instead of editing all of the server's separate configuration files.

## Making a substitution

To support the symbol-to-string mappings, create a file named substitution.xml. This file's name must be all lowercase, since filenames are case-sensitive on Linux-based operating systems, but not on Windows platforms.

The substitution.xml file is a place to describe the symbol-to-string mappings that you defined. The Flash Media Server 2 installer also defines a few of these mappings during the installation process, and it stores them in a separate file called fms.ini. The server looks for both of these files in the conf directory located at the same level as the Flash Media Server executable files. You can also choose to define these mappings in other files that you create, and create references to these custom files in the substitution.xml file.

The substitution.xml file defines text-to-symbol mappings using the following form:

```
<Root>
  <Symbols>
    <SymbolName>StringToMapTo</SymbolName>
    <SymbolName>StringToMapTo</SymbolName>
  </Symbols>
</Root>
```

Within the `Symbols` tag, create child tags that describe the symbol names. Place the string to be mapped to the symbol within the opening and closing child tag.

For example, this tag maps the symbol `VIR_DIR` to the string `"c:\streams"`:

```
<VIR_DIR>c:\streams</VIR_DIR>
```

In the substitution.xml file, this tag would be a child of the `Symbols` tag, as in the following XML fragment:

```
<Root>
  <Symbols>
    <VIR_DIR>c:\streams</VIR_DIR>
  </Symbols>
</Root>
```

Once you have defined such a mapping, you can use the symbol in one of the XML configuration files.

To use a symbol in place of a normal string in a configuration file, specify the symbol name, with the characters `${` before the symbol name, and `}` after the symbol name. Whenever the server finds something of the form `${SYMBOL}`, it performs a lookup to see if the symbol is mapped to a string. If no mapping is found, then `${SYMBOL}` is not substituted, and is taken literally. Otherwise, it is substituted.

For example, in the Vhost.xml file, you might use the previously defined symbol as follows:

```
<VirtualHost>
  <VirtualDir>
    <Streams>foo;${VIR_DIR}</Streams>
  </VirtualDir>
</VirtualHost>
```

When the server encounters the symbol, it checks whether the symbol named `VIR_DIR` is mapped to anything. It then finds that it is mapped to `c:\streams` in the substitution.xml file. The symbolic mapping in the previous XML fragment is equivalent to the following XML without symbols:

```
<VirtualHost>
  <VirtualDir>
    <Streams>foo;c:\streams</Streams>
  </VirtualDir>
</VirtualHost>
```

If the substitution.xml file is missing and you try to use text substitution symbols in the configuration files, the server will interpret the symbols as literal strings.

## Predefined symbols

The two predefined symbols `ROOT` and `CONF` do not need to be mapped, and are always available. The `ROOT` symbol is mapped to the location of the FMSMaster.exe file, as this example shows:

```
<AppsDir>${ROOT}\..\..\myapps</AppsDir>
```

The `CONF` symbol is mapped to the location of the Server.xml file, as this example shows:

```
<AppsDir>${CONF}\..\yourapps</AppsDir>
```

## Mapping environment variables

You can also specify symbols that resolve to environment variables. To refer to an environment variable in one of the XML configuration files, use the name of the environment variable within percent (%) characters. The % characters indicate to the server that the symbol refers to an environment variable, and not to a user-defined string.

The syntax for specifying an environment variable as a symbol is `${%ENV_VAR_NAME%}`.

For example, the server will map the following symbol to the `COMPUTERNAME` variable:

```
${%COMPUTERNAME%}
```

## Defining symbols outside the substitution.xml file

You can specify all of your text substitution mappings under the `Symbols` tag in subsitutions.xml. However, you can also specify separate external files that contain symbol-to-string mappings. To do this, specify one or more `KeyValueFile` tags in the substitution.xml file. Each of these tags can specify the location of one external file.

For example, the following XML specifies the file C:\testfiles\mySymbols.txt within the substitution.xml file:

```
<Root>
  <KeyValueFile>C:\testfiles\mySymbols.txt</KeyValueFile>
</Root>
```

These external files are not in XML format. They simply contain a collection of symbol-value pairs, where each symbol-value pair appears on a separate line and takes the following form:

*`symbol=value`*

The following example shows three symbol-value pairs:

```
USER_NAME=foo
USER_PSWD = bar
HELLO= " world "
```

Place comments on separate lines that begin with a number sign (#). Do not place comments on the same line as a symbol definition.

The first equal sign (=) found in a line is considered the delimiter that separates the symbol and the value. The server trims any leading or trailing white space from both the symbol and the value. If the value is surrounded by double quotation marks, the server does not trim leading or trailing white space within the quotation marks.

## Building the symbol map

You may use a symbol anywhere, such as in the substitution.xml file, or any of the external configuration files, as long as it has been defined before the server encounters it. The server builds the symbol map in the following order:

1. The predefined symbols `ROOT` and `CONF` are evaluated first.
2. The fms.ini file is evaluated next.
3. If the substitution.xml file exists, the server looks for the `Symbols` tag and processes the child tags in the order in which they appear.
4. The server finds each `KeyValueFile` tag, and processes the specified files in the order in which they appear. Symbols defined in these external configuration files are processed in the order in which they appear in each file.

# Configurable application object properties for server-side scripting

Flash Media Server 2 supports configuration tags that enhance the server-side application object. You can now define properties for the application object within the server's XML configuration files.

To define properties of the application object, specify the custom configuration tags in the `JSEngine` section of the Application.xml file. The property name corresponds to the tag's name, and the property value corresponds to the tag's contents.

For example, the following XML fragment defines the properties `user_name` and `dept_name`, with the values `jdoe` and `engineering`, respectively:

```
<Application>
  <JSEngine>
       <config>
         <user_name>jdoe</user_name>
         <dept_name>engineering</dept_name>
       </config>
  </JSEngine>
</Application>
```

To access the property in server-side code, use the syntax in either of these examples:

```
application.config.prop_name
application.config["prop_name"]
```

For example, given the previous XML fragment, the following `trace()` statements are valid:

```
trace("I am " + application.config.user_name + " and I work in the " +
  application.config.dept_name + " department.");
trace("I am " + application.config["user_name"] + " and I work in the " +
  application.config["dept_name"] + " department.");
```

The output from either statement would be as follows:

```
I am jdoe and I work in the engineering department.
```

As a second example, assume that the environment variable `COMPUTERNAME` is equal to `"jsmith01"`, and you have defined a symbol named `HELLO` in the substitution.xml file, as follows:

```
<Root>
  <Symbols>
    <HELLO>World</HELLO>
  </Symbols>
</Root>
```

In addition, the following XML appears in the Application.xml file:

```
<Application>
  <JSEngine>
       <config>
         <foo>${%COMPUTERNAME%}</foo>
         <hello>${HELLO}</hello>
       </config>
  </JSEngine>
</Application>
```

Then in a server-side script, the following `trace()` statements would be valid:

```
trace("My computer's name is: " + application.config.foo);
trace("Hello " + application.config.hello);
```

The output would then be as follows:

```
My computer's name is: jsmith01
Hello World
```

**NOTE** In Flash Media Server 2, the output of `trace()` statements is displayed in the management console and the application log file.

CHAPTER 3

# Configuration Files

3

This chapter describes the XML files that define the Flash Media Server configuration. It presents their file structures, a summary of the tags in each file, and detailed information about the tags in the configuration files.

Macromedia Flash Media Server 2 accommodates a wide range of applications. After installation, the server's configuration files contain generic settings. As you develop and test your applications, you will use these files to configure the server to best suit these applications.

## XML configuration files

Flash Media Server uses separate XML configuration files for each level of the server hierarchy: server, adaptor, virtual host, application, and logging. Each of these files contains configuration tags that relate to the server, adaptor, virtual host, application, or logging activity they are associated with. To customize the functionality of the server, you edit these tags. The server edits some of these tags itself when you use the management console.

When you're ready to customize the server for your own virtual hosts and applications, you'll edit the server's XML configuration files and the directory structure that contains them. Flash Media Server uses the following configuration files in XML format:

- Server.xml
- Users.xml
- Logger.xml
- Adaptor.xml
- Vhost.xml
- Application.xml

You configure Flash Media Server by editing the contents of these files, either in a text editor or with the management console. If you edit any XML configuration files, you must save the files and then restart the server before the new settings take effect.

The following sections describe the tags in each XML file in detail.

# Server.xml file

The Server.xml file is located at the root level of the `conf` directory and contains the tags and information used to configure Flash Media Server 2. You can edit the Server.xml file to add or remove configuration information.

The Server.xml file contains the following tag structure.

```
<Root>
    <Server>
      <AutoDiscovery>
        <Enable type="">false</Enable>
        <BindInfo></BindInfo>
        <ProxyInfo></ProxyInfo>
        <SecureProxyInfo></SecureProxyInfo>
        <Allow></Allow>
        <Deny></Deny>
        <Order>Allow,Deny</Order>
        <AllowZones></AllowZones>
        <MyZone></MyZone>
        <TTL>1</TTL>
        <ClusterMonitorInterval>60</ClusterMonitorInterval>
        <BroadcastAddress>255.255.255.255</BroadcastAddress>
        <BroadcastPort>67</BroadcastPort>
        <MaxWaitTime>100</MaxWaitTime>
        <UserData></UserData>
      </AutoDiscovery>
      <SSL>
        <SSLRandomSeed>16</SSLRandomSeed>
        <SSLSessionCacheGC>5</SSLSessionCacheGC>
        <SSLClientCtx>
          <SSLVerifyCertificate>true</SSLVerifyCertificate>
          <SSLCACertificatePath></SSLCACertificatePath>
          <SSLCACertificateFile></SSLCACertificateFile>
          <SSLVerifyDepth>9</SSLVerifyDepth>
          <SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH</
  <SSLCipherSuite>
        </SSLClientCtx>
      </SSL>
      <Process>
        <UID>${SERVER.PROCESS_UID}</UID> [Linux only]
        <GID>${SERVER.PROCESS_GID}</GID> [Linux only]
      </Process>
      <Mask>017</Mask>
      <AdminServer>
        <HostPort>${SERVER.ADMINSERVER_HOSTPORT}</HostPort>
        <SocketGC>60</SocketGC>
        <Process>
          <UID>${SERVER.PROCESS_UID}</UID> [Linux only]
          <GID>${SERVER.PROCESS_GID}</GID> [Linux only]
```

```
  </Process>
</AdminServer>
<ResourceLimits>
  <CPUMonitor>1</CPUMonitor>
  <ThreadPoolGC>20</ThreadPoolGC>
  <MsgPoolGC>60</MsgPoolGC>
  <ApplicationGC>5</ApplicationGC>
  <FLVCacheSize>40</FLVCacheSize>
  <SocketGC>60</SocketGC>
  <SSLSessionCacheGC>5</SSLSessionCacheGC>
  <Connector>
    <HTTP>
      <MinIOThreads>0</MinIOThreads>
      <MaxIOThreads>0</MaxIOThreads>
      <MaxConnectionQueueSize>-1</MaxConnectionQueueSize>
    </HTTP>
    <RTMP>
      <MinIOThreads>0</MinIOThreads>
      <MaxIOThreads>32</MaxIOThreads>
      <NumCRThreads>0</NumCRThreads>
      <MinConnectionThreads>0</MinConnectionThreads>
      <MaxConnectionThreads>0</MaxConnectionThreads>
      <MaxConnectionQueueSize>-1</MaxConnectionQueueSize>
    </RTMP>
  </Connector>
  <Protocol>
    <RTMP>
      <Edge>
        <MinIOThreads>0</MinIOThreads>
        <MaxIOThreads>0</MaxIOThreads>
        <SocketTableSize>-1</SocketTableSize>
        <SocketOverflowBuckets>-1</SocketOverflowBuckets>
      </Edge>
      <Core>
        <MinIOThreads>0</MinIOThreads>
        <MaxIOThreads>0</MaxIOThreads>
        <SocketTableSize>-1</SocketTableSize>
        <SocketOverflowBuckets>-1</SocketOverflowBuckets>
      </Core>
      <Admin>
        <MinIOThreads>0</MinIOThreads>
        <MaxIOThreads>0</MaxIOThreads>
        <SocketTableSize>-1</SocketTableSize>
        <SocketOverflowBuckets>-1</SocketOverflowBuckets>
      </Admin>
    </RTMP>
    <ECCP>
        <MinIOThreads>0</MinIOThreads>
        <MaxIOThreads>0</MaxIOThreads>
        <SocketTableSize>-1</SocketTableSize>
```

```
        <SocketOverflowBuckets>-1</SocketOverflowBuckets>
        <CoreTimeout>30</CoreTimeout>
    </ECCP>
    <ACCP>
        <MinIOThreads>0</MinIOThreads>
        <MaxIOThreads>0</MaxIOThreads>
        <SocketTableSize>-1</SocketTableSize>
        <SocketOverflowBuckets>-1</SocketOverflowBuckets>
    </ACCP>
  </Protocol>
  <IPCQueues>
    <GlobalQueue>
      <HeapSize>1024</HeapSize>
      <MaxQueueSize>100</MaxQueueSize>
    </GlobalQueue>
    <EdgeCore>
      <HeapSize>2048</HeapSize>
      <MaxQueueSize>100</MaxQueueSize>
    </EdgeCore>
    <Services>
      <HeapSize>2048</HeapSize>
      <MaxQueueSize>100</MaxQueueSize>
    </Services>
  </IPCQueues>
  <MessageCache>
    <MaxCacheUnits>4096</MaxCacheUnits>
    <MaxCacheSize>100</MaxCacheSize>
    <MaxUnitSize>16</MaxUnitSize>
    <FreeRatio>0.125</FreeRatio>
    <GlobalRatio>0.4</GlobalRatio>
    <MaxAge>1000000</MaxAge>
    <UpdateInterval>1024</UpdateInterval>
    <FreeMemRatio>0.5</FreeMemRatio>
  </MessageCache>
  <SmallMemPool>
    <MaxCacheUnits>4096</MaxCacheUnits>
    <MaxCacheSize>100</MaxCacheSize>
    <MaxUnitSize>16</MaxUnitSize>
    <FreeRatio>0.125</FreeRatio>
    <GlobalRatio>0.4</GlobalRatio>
    <MaxAge>1000000</MaxAge>
    <UpdateInterval>1024</UpdateInterval>
    <FreeMemRatio>0.5</FreeMemRatio>
  </SmallMemPool>
  <LargeMemPool>
    <MaxCacheUnits>4096</MaxCacheUnits>
    <MaxCacheSize>100</MaxCacheSize>
    <MaxUnitSize>16</MaxUnitSize>
    <FreeRatio>0.125</FreeRatio>
    <GlobalRatio>0.4</GlobalRatio>
```

```
            <MaxAge>1000000</MaxAge>
            <UpdateInterval>1024</UpdateInterval>
            <FreeMemRatio>0.5</FreeMemRatio>
          </LargeMemPool>
          <SegmentsPool>
            <MaxCacheUnits>4096</MaxCacheUnits>
            <MaxCacheSize>100</MaxCacheSize>
            <MaxUnitSize>16</MaxUnitSize>
            <FreeRatio>0.125</FreeRatio>
            <GlobalRatio>0.4</GlobalRatio>
            <MaxAge>1000000</MaxAge>
            <UpdateInterval>1024</UpdateInterval>
            <FreeMemRatio>0.5</FreeMemRatio>
          </SegmentsPool>
          <Master>
            <CoreGC>300</CoreGC>
            <CoreExitDelay>20</CoreExitDelay>
          </Master>
        </ResourceLimits>
        <Logging>
          <Time>local</Time>
          <Access>
            <Enable>true</Enable>
            <Scope>server</Scope>
          </Access>
          <Diagnostic>
            <Enable>true</Enable>
          </Diagnostic>
          <Application>
            <Enable>true</Enable>
          </Application>
        </Logging>
        </HttpProxy>
        <LocalHost>127.0.0.1</LocalHost>
      </Server>
      <ServerDomain></ServerDomain>
</Root>
```

# Summary of Server.xml tags

This table lists alphabetically the tags in the Flash Media Server Server.xml configuration file.

| Server.xml tag | Description |
|---|---|
| Access | Container tag; contains the tags to configure the Access log settings. |
| ACCP | Container tag; contains tags to configure the Admin core communication protocol (ACCP). |

| Server.xml tag | Description |
|---|---|
| Admin | Container tag; contains the tags that configure the RTMP protocols for the FMSAdmin.exe process. |
| AdminServer | Container tag; contains tags to configure the Flash Media Admin Service. |
| Allow | Specifies which automatic proxy discovery messages Flash Media Server responds to. |
| AllowZones | Specifies which clients this proxy server will respond to with the Autodiscovery message. |
| Application | Container tag; the Enable tag in this container enables or disables the Application log file. |
| ApplicationGC | Specifies how often to check for and remove unused applications. |
| AutoDiscovery | Container tag; contains tags to configure the Flash Media Server automatic proxy discovery process. |
| BindInfo | Specifies the IP and port that FMSp listens on for automatic proxy discovery messages. |
| BroadcastAddress | Specifies the broadcast address to use for broadcasting FPAD messages. |
| BroadcastPort | Specifies the broadcast port to use for broadcasting FPAD messages. |
| ClusterMonitorInterval | Specifies in seconds how often to check for edges in the cluster that have not sent a keep-alive message. |
| Connector | Container tag; contains tags to configure the connector subsystem. Flash Media Server provides connectors that allow application scripts to connect to other Flash Media Servers or HTTP servers. |
| Core | Container tag; contains tags to configure the protocols for the FMSCore.exe process. |
| CoreExitDelay | Specifies the wait time for an idle core to exit on its own before it is removed from the server. |
| CoreGC | Specifies how often, in seconds, to check for and remove idle cores. |
| CoreTimeout | Specifies the time-out value for detecting unresponsive cores. |
| CPUMonitor | Specifies how often to monitor CPU usage. |

| Server.xml tag | Description |
|---|---|
| Deny | Specifies which automatic proxy discovery messages not to respond to. |
| Diagnostic | Container tag; contains tag to enable the diagnostic log file. |
| ECCP | Container tag; contains tags to configure the edge core communication protocol. |
| Edge | Container tag; contains tags to configure the RTMP protocol for FMSEdge.exe process. |
| EdgeCore | Container tag; these tags control the IPC message queues used by edge and core processes to communicate with each other. |
| Enable (Access) | Enables or disables the Access logs. |
| Enable (Application) | Enables or disables the Application logs. |
| Enable (AutoDiscovery) | Enables or disables the Flash Media Server automatic proxy discovery process. |
| Enable (Diagnostic) | Enables or disables the diagnostic logs. |
| FLVCacheSize | Specifies the maximum size of the FLV cache. |
| FreeMemRatio | Sets the maximum percentage of total memory that the total pool size may occupy. |
| FreeRatio | Specifies the percentage of the message cache to be consumed by the free list on a per-thread basis. |
| GID | Contains the group ID of the server process. |
| GlobalQueue | Container tag; these tags control the IPC message queue used by processes to communicate with each other. |
| GlobalRatio | Specifies the percentage of the message cache that can be consumed by the free list on a global basis. |
| HeapSize | Specifies the maximum size of the shared memory heap used for a IPC message queue. |
| HostPort | Specifies the IP address and port that the Flash Media Admin Service binds to. |
| HTTP | Container tag; contains tags to configure the HTTP connector, which is used by remote sites for accessing Flash Media Server. |
| IPCQueues | Container tag; contains tags to configure the IPC (interprocess communication) queues |

| Server.xml tag | Description |
|---|---|
| LargeMemPool | Container tag; contains tags to configure the large memory pool. |
| LocalHost | Specifies the Flash Media Server IP loopback address. |
| Logging | Container tag; contains tags to perform the overall logging configuration. |
| Mask | Contains a three-digit octal value used by the Linux `umask` (user permissions mask) command to set a file creation mask. |
| Master | Container tag; contains tags to configure the resource limits for the master server. |
| MaxAge | Specifies the maximum reuse count before freeing the cache unit. |
| MaxCacheSize | Specifies the maximum size of the cache. |
| MaxCacheUnits | Specifies the maximum free units in the cache. |
| MaxConnectionQueueSize | Specifies the maximum number of connection requests that can be pending. |
| MaxConnectionThreads | Specifies the maximum number of threads used to process connection requests. |
| MaxIOThreads | Specifies the maximum number of threads that can be created for I/O processing. |
| MaxQueueSize | Specifies the maximum size of the queue in messages. |
| MaxUnitSize | Sets the threshold of the maximum message size to get back into the cache. |
| MaxWaitTime | Defines the maximum time in milliseconds that the client should wait for additional autodiscovery responses from other proxy servers. |
| MessageCache | Container tag; contains tags to control how the message cache keeps messages used by Flash Media Server. |
| MinConnectionThreads | Specifies the minimum number of threads in the pool for I/O operations. |
| MinIOThreads | Specifies the minimum number of threads that can be created for I/O operations. |
| MsgPoolGC | Specifies how often Flash Media Server checks for and removes content in the global message pool. |

| Server.xml tag | Description |
|---|---|
| MyZone | Specifies the zone that the edge server belongs to when it broadcasts FPAD messages. |
| NumCRThreads | Specifies the number of completion routine threads for edge server I/O processing on WIndows 32-byte systems. |
| Order | Specifies whether the Allow or Deny tag is evaluated first. |
| Process | Container tag: contains the ID tags for all server processes on Linux. |
| Protocol | Container tag; contains tags to configure protocols and their reception. |
| ProxyInfo | Specifies the host (or IP address) and port number to return to the client in the FPAD response. |
| ResourceLimits | Container tag; contains tags to specify the maximum resource limits for the server. |
| Root | Root tag: contains all other tags in Server.xml. |
| RTMP (Connector) | Container tag; contains tags to configure the RTMP connector. |
| RTMP (Protocol) | Container tag; contains tags to configure the RTMP protocol. |
| Scope | Determines whether or not to write a log file for each virtual host or write only one log file for the server. |
| SecureProxyInfo | Specifies the host (or IP address) and port number to return to client in the FPAD response for clients wanting to make a secure connection. |
| SegmentsPool | Container tag; contains tags that configure how the segments pool caches segments of FLV files. |
| Server | Container tag; contains tags that configure the server. |
| ServerDomain | Specifies the host name (with domain) of the server machine. |
| Services | Container tag; contains tags to control the IPC message queue used by edge and core processes to communicate with each other. |
| SmallMemPool | Container tag; contains tags to configure the small memory pool. |
| SocketGC | Specifies how often to check for and remove inactive sockets. |

| Server.xml tag | Description |
|---|---|
| SocketOverflowBuckets | Specifies the number of overflow buckets if all slots in socket table are in use. |
| SocketTableSize | Specifies the size of the direct access socket table for quick lookup. |
| SSL | Container tag; contains tags to configure Flash Media Server as an SSL-enabled client for secure communications. |
| SSLCACertificateFile | Specifies the name of a file that contains one or more CA certificates in PEM encryption format. |
| SSLCACertificatePath | Specifies the name of the directory containing one or more CA certificates. |
| SSLCipherSuite | Specifies the encryption ciphers to secure outgoing communications. |
| SSLClientCtx | Container tag; contains tags to configure Flash Media Server as an SSL (Secure Socket Layer) client for outgoing SSL connections. |
| SSLRandomSeed | Specifies the number of bytes of entropy to use for seeding the pseudo-random number generator (PRNG). |
| SSLSessionCacheGC | Specifies how often to flush expired sessions from the server-side SSL session cache. |
| SSLVerifyCertificate | Specifies whether or not to verify the certificate returned by the server being connected to. |
| SSLVerifyDepth | Specifies the maximum depth in the certificate chain that Flash Media Server is willing to accept. |
| ThreadPoolGC | Specifies how often to check for and remove unused I/O threads. |
| Time | Specifies the time field in a log file. |
| TTL | Specifies in minutes how often to broadcast a keep-alive message to other edges in the cluster, and how often another edge should expect to receive a keep-alive message from this edge. |
| UID | Contains the server process user ID. |
| UpdateInterval | Specifies how often thread statistics are collected. |
| UserData | Specifies the user data that is returned to the client through the FPAD response. |

# Description of Server.xml tags

The following alphabetical list of Server.xml tags contains additional information, including cross references to associated tags, syntax, and examples.

## Access

Container tag.

### Description

The tags nested within the `Access` container configure the Access log settings.

### Contained tags

`Enable (Access)`, `Scope`

## ACCP

Container tag.

### Description

The tags nested within the `ACCP` container configure the Admin Core Communication Protocol (ACCP). The Flash Media Admin Service and active cores use ACCP for communications. This protocol is also used for collecting performance metrics and issuing administrative commands to Flash Media Server cores.

The Admin Service is separate from the Flash Media Server. When administrators connect to the server with the management console, they are connecting to the Flash Media Admin Service, which in turn connects to the Flash Media Server.

### Contained tags

`MinIOThreads`, `MaxIOThreads`, `SocketTableSize`, `SocketOverflowBuckets`

### See also

`Admin`, `Core`, `ECCP`, `Edge`, `HTTP`, `RTMP (Protocol)` containers

## Admin

Container tag.

### Description

The tags nested within the `Admin` container configure the RTMP (Real-Time Messaging Protocol) for the FMSAdmin.exe process. RTMP is the protocol used for communication between end users (typically users using Flash Player) and Flash Media Server.

### Contained tags

MinIOThreads, MaxIOThreads, SocketOverflowBuckets, SocketTableSize

### See also

ACCP, Core, ECCP, Edge, HTTP, RTMP (Protocol) containers

## AdminServer

Container tag.

### Description

The tags nested within the AdminServer container configure the Flash Media Admin Service.

### Contained tags

HostPort, SocketGC, Process, UID, GID

## Allow

This tag identifies those computers that broadcast automatic proxy discovery messages that the Flash Media Server responds to.

### Description

The Allow tag is a comma-delimited list of host names, domain names, and full or partial IP address, as well as the keyword all.

### Example

```
<Allow>x.foo.com, foo.com, 10.60.1.133, 10.60</Allow>
```

or

```
<Allow>all</Allow>
```

These examples list the computers sending requests that Flash Media Server will process.

| NOTE | Macromedia does not recommend the use of "all" as an attribute. It creates an opportunity for a security risk. |
|---|---|

### See also

Deny, Order

## AllowZones

This tag specifies which clients this server will respond to with the Autodiscovery message.

### Description

This tag is a comma-delimited list of zones that the sole origin server or the edge servers in a cluster will service. While the `Allow` and `Deny` tags restrict access based on IP address or host name, the `AllowZones` tag allows access based on the zone where the client is located.

A zone is a number, and a client is assigned as belonging to a particular zone by setting the `NetConnection.fpadZone` property in the `NetConnection.connect()` API.

By default, clients belong to zone 0. If this tag is left empty, Flash Media Server services all zones.

When this tag is set to Zone 2 but it receives a message from a client in Zone 1, the server does not reply to this client.

### Example

```
<AllowZones>1,3,5</AllowZones>
```

This example configures the proxy server to respond to requests from clients in zones 1, 3, and 5, but not from clients in zones 2 and 4.

### See also

`Allow`, `Deny`, `Order`

## Application

Container tag.

### Description

The `Enable` tag nested within the `Application` container enables the Application log file.

### Contained tag

`Enable (Access)`

## ApplicationGC

This tag specifies in minutes how often Flash Media Server checks for and removes unused application instances.

### Description

The default interval is 5 minutes, which is also the minimum value for this tag.

## AutoDiscovery

Container tag.

### Description

The tags nested within the AutoDiscovery container set up and configure a single, or a cluster of, edge or proxy servers.

### Contained tags

Allow, AllowZones, BindInfo, BroadcastAddress, BroadcastPort, ClusterMonitorInterval, Deny, Enable (AutoDiscovery), MyZone, Order, ProxyInfo, SecureProxyInfo, TTL

## BindInfo

This tag identifies the IP and port number that Flash Media Server listens on for proxy autodiscovery messages.

### Description

If no IP address is specified, Flash Media Server listens on the port specified by the Windows system variable INADDR_ANY.

By default, Flash Media Server listens on any available interface on port 67.

| NOTE | If this computer has multiple interfaces, an automatic proxy discovery message will be received by each available interface unless you specify a specific interface to bind to. |
|---|---|

### Syntax

```
<BindInfo>[ip]:<port></BindInfo>
```

### See also

ProxyInfo in this container.

## BroadcastAddress

This tag specifies the address to use when broadcasting FPAD (Flash Proxy Auto-Discovery) messages.

### Description

The default address is 255.255.255.255.

## BroadcastPort

This tag specifies the port to use when broadcasting FPAD messages.

### Description

The default port is 67. This is a DHCP (Dynamic Host Configuration Protocol) port.

DHCP is a protocol for assigning dynamic IP addresses to devices on a network. DHCP supports a mix of static and dynamic IP addresses.

## ClusterMonitorInterval

This tag specifies in seconds how often to check for *stale* edges.

### Description

Stale edges are those edges that have not sent the FADP a keep-alive message within the specified time limit.

The default value is 60 seconds.

### See also

`TTL`

## Connector

Container tag.

### Description

The tags nested within the `Connector` container configure the connector subsystem. Flash Media Server provides connectors that allow application scripts to connect to other Flash Media Servers or HTTP servers.

### Contained tags

`HTTP`, `RTMP (Connector)`

### See also

`RTMP (Protocol)` in the `Protocol` container

## Core

Container tag.

### Description

The tags nested within the `Core` container configure the RTMP protocol for the FMSCore.exe process.

### Contained tags

`MinIOThreads`, `MaxIOThreads`, `SocketOverflowBuckets`, `SocketTableSize`

### See also

ACCP, Admin, ECCP, Edge, HTTP, RTMP (Protocol) containers

## CoreGC

This tag specifies how often to check for and remove idle or unused cores.

### Description

The default is 300 seconds.

## CoreTimeout

This tag specifies the timeout value for detecting unresponsive cores.

### Description

The default timeout is 30 seconds. A value of 0 disables the timeout check.

## CoreExitDelay

This tag specifies how much wait time an idle core is given to exit on its own before it is removed from the server.

### Description

The default wait time is 20 seconds.

## CPUMonitor

This tag specifies in seconds how often Flash Media Server monitors CPU usage.

### Description

The default interval is 1 second. The value cannot be set to less than 1 second.

## Diagnostic

Container tag.

### Description

The `Enable` tag nested within the `Diagnostic` section enables the diagnostic log file.

### Contained tag

`Enable (Access)`

## Deny

This tag specifies which automatic proxy discovery messages Flash Media Server does not respond to.

### Description

This tag is a comma-delimited list of host names, domain names, and full or partial IP address, as well as the keyword `all`. This tag works in conjunction with the `Allow` and `Order` tags to determine which automatic proxy discovery messages Flash Media Server responds to.

### Examples

```
<Deny>x.foo.com, foo.com, 10.60.1.133, 10.60</Deny>
<Deny>all</Deny>
```

### See also

`Allow`, `Order` tags

## ECCP

Container tag.

The tags nested within the `ECCP` container configure ECCP [Edge Server-Core Server Communication Protocol].

### Description

Flash Media Server edge processes and Flash Media Server core processes use ECCP to migrate socket connections and proxy non-migrated connections.

### Contained tags

`CoreTimeout`, `MinIOThreads`, `MaxIOThreads`, `SocketOverflowBuckets`, `SocketTableSize`

### See also

`ACCP`, `Admin`, `Core`, `Edge`, `HTTP`, `RTMP (Protocol)` containers

## Edge

Container tag.

### Description

The tags nested within the `Edge` container configure the RTMP protocol for the FMSEdge.exe process.

### Contained tags

MinIOThreads, MaxIOThreads, SocketOverflowBuckets, SocketTableSize

### See also

ACCP, Admin, Core, ECCP, HTTP, RTMP (Protocol) containers

## EdgeCore

Container tag.

### Description

The tags nested within the EdgeCore container control the IPC (interprocess communication) message queue used by edge and core processes to communicate with each other.

### Contained tags

HeapSize, MaxCacheSize

## Enable (Access)

Server.xml uses four tags named Enable: the Enable tag in the AutoDiscovery container and the Enable tags in the Access, Application, and Diagnostic subdirectories in the Logging container.

Located in the Logging container.

### Description

This tag enables or disables the access logs. A value of true enables the logging process; false disables the logging process. The default setting is true.

## Enable (Application)

Server.xml uses four tags named Enable: the Enable tag in the AutoDiscovery container and the Enable tags in the Access, Application, and Diagnostic subdirectories in the Logging container.

Located in the Logging container.

### Description

This tag enables or disables the application logs. A value of true enables the logging process; false disables the logging process. The default setting is true.

## Enable (AutoDiscovery)

Server.xml uses four tags named `Enable`: the `Enable` tag in the `AutoDiscovery` container and the `Enable` tags in the Access, Application, and Diagnostic subdirectories in the `Logging` container.

### Description

This tag enables or disables the Flash Media Server automatic proxy discovery process. A value of `true` enables the process; `false` disables the process. If the `Enable` tag is left unspecified, the automatic proxy discovery process is disabled.

## Enable (Diagnostic)

Server.xml uses four tags named `Enable`: the `Enable` tag in the `AutoDiscovery` container and the `Enable` tags in the Access, Application, and Diagnostic subdirectories in the `Logging` container.

Located in the `Logging` container.

### Description

This tag enables or disables the diagnostic logs. A value of `true` enables the logging process; `false` disables the logging process. The default setting is `true`.

## FLVCacheSize

This tag specifies the maximum size of the FLV (Flash Video) cache.

### Description

The FLV cache size is specified as a percentage of the total physical memory on the system. The default setting for cache size is 40 (40 percent). The maximum setting is 100 (100 percent).

## FreeMemRatio

Located in the `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

### Description

This tag specifies the maximum percentage of total memory that the total pool size may occupy. The range of this setting lies between 0 (0 percent) and 1 (100 percent). The default setting is 0.5 (50 percent).

### See also

FreeRatio

## FreeRatio

Located in the LargeMemPool, MessageCache, SegmentsPool, and SmallMemPool containers.

### Description

This tag specifies the percentage of the message cache to be consumed by the free list on a per-thread basis. The range of this setting lies between 0 (0 percent) and 1 (100 percent). The default setting is 0.125 (12.5 percent).

When more free memory is available to a thread than the specified ratio, the freed memory will return to the global pool.

### See also

FreeMemRatio

## GID

This tag specifies the group ID of the server process.

### Description

This tag is applicable for Flash Media Server running on Linux systems only.

### See also

UID

## GlobalQueue

Container tag.

### Description

The tags nested within the GlobalQueue container control the IPC message queue used by all processes to communicate with each other.

### Contained tags

HeapSize, MaxQueueSize

## GlobalRatio

Located in the `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

### Description

This tag specifies the percentage of the message cache to be consumed by the free list on a global basis. When more free memory is available to a thread than the specified ratio, the freed memory will return to the operating system.

The range of this setting lies between 0 (0 percent) and 1 (100 percent). The default setting is 0.4 (40 percent).

### See also

`FreeMemRatio`, `FreeRatio`

## HeapSize

Located in the `EdgeCore`, `GlobalQueue`, and `Services` containers.

### Description

This tag specifies the maximum size in kilobytes of the shared memory heap used for an IPC (interprocess communication) message queue. The default value for this tag varies according to its container.

| Container | Default Value | Description |
|---|---|---|
| `EdgeCore` | 1024 | If the maximum size for this tag is not specified, the value is 100Kb. |
| `GlobalQueue` | 2048 | |
| `Services` | 2048 | |

## HostPort

This tag specifies the IP address and port number that the Flash Media Admin Service binds to.

### Description

The default is to bind to any available IP on port 1111. Only one port number may be specified in this tag.

The Admin Service is separate from the Flash Media Server. When administrators connect to the server with the management console, they are connecting to the Flash Media Admin Service, which in turn connects to the Flash Media Server.

### Syntax

```
<HostPort>[<ip>][:<port>]</HostPort>
```

## HTTP

Container tag.

### Description

The tags nested within the `HTTP` container configure the HTTP connector, which is used by remote Flash Player sites to access Flash Media Server.

The following reference table gives the default values for all thread configurations.

| Default Value | Description |
|---|---|
| 0 | Allocates the default number of threads. |
| >0 | Allocates the exact number of threads specified. |
| >0 | Associates the default value with the number (N) of processors. |
| -1 | Allocates 1xN threads. |
| -2 | Allocates 2xN threads. |

### Contained tags

`MinIOThreads`, `MaxIOThreads`, `SocketOverflowBuckets`, `SocketTableSize`

## IPCQueues

Container tag.

### Description

The tags nested within the `IPCQueues` container configure the IPC queues. Flash Media Server uses IPC queues to send messages from one core to another or from one process to another, such as master to core, or core to edge.

Unlike protocols, queues are used for short or one-time messages that may have more than one target.

### Contained tags

`HeapSize`, `MaxQueueSize`

## LargeMemPool

Container tag.

### Description

The tags nested within the `LargeMemPool` container configure the large memory pool, which caches large chunks of memory within Flash Media Server to increase performance of large allocations.

### Contained tags

`FreeMemRatio`, `FreeRatio`, `GlobalRatio`, `MaxAge`, `MaxCacheSize`, `MaxUnitSize`, `UpdateInterval`

### See also

`MessageCache`, `SegmentsPool`, `SmallMemPool` containers

## LocalHost

Specifies the Flash Media Server IP loopback address.

### Description

Flash Media Server must reference itself locally. The IP loopback address is usually the default 127.0.0.1 address. With more than one network interface, 127.0.0.1 can map to an erroneous interface. The server will use the default loopback address as the local loopback.

## Logging

Container tag.

### Description

The tags nested within the `Logging` container perform the overall logging configuration. You set the configuration properties of the individual log files in the Logger.xml file.

Log files are written in English. Field names in the log file are in English. Some content within the log file, however, may be in another language, depending on the filename and the operating system. For example, in the Access.log file, the columns x-sname and x-suri-stem show the name of the stream. If the name of the recorded stream is in a language other than English, the stream's name will be written in the log file in that language, even if the server is running on an English-language operating system.

### Contained tags

`Time`, `Access`, `Diagnostic`, `Application` containers

### See also

Logger.xml file

## Mask

A three-digit octal value used by the Linux `umask` (user permissions mask) command to set a file creation mask. The user must enter the mask in a three-digit octal format.

The default setting for this tag is `017` in octal.

### Description

This tag is applicable for Flash Media Server running Linux systems only. This tag controls who has read/write access to shared object and stream files in the server. All Flash Media Server object files, such as stream files or shared object files, are created on the server side with permission `0666`. This key is used by `umask` to set the file creation mask. By default, the creation mask is set to `017` in octal. Therefore, all the Flash Media Server object files are created with permission `0666 & ~017 = 0660 = rw-rw----`.

The owner and the users who belong to the same group as the owner will get read/write permission to the files. If the mask is set to `022`, the file created will have permission `0666 & ~022 = 0644 = rw-r--r--`.

## Master

Container tag.

### Description

The tags nested within the `Master` container configure the resource limits for the master server.

### Contained tags

`CoreGC`, `CoreExitDelay`

## MaxAge

Located in the `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

### Description

This tag defines the maximum reuse count before the cache unit is freed. The default count is 1,000,000.

## MaxCacheSize

Located in the `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

### Description

This tag defines the maximum size of the cache in megabytes. The default is 100 MB.

### See also

`MaxCacheUnits`

## MaxCacheUnits

Located in the `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

### Description

This tag defines the maximum number of free units in the cache. Keep in mind that the number of free units may be less than maximum if the value of the `MaxCacheSize` limit is reached.

The default is 4096 units.

### See also

`MaxCacheSize`

## MaxConnectionQueueSize

Located in the `HTTP` and `RTMP (Connector)` containers.

### Description

This tag specifies the maximum number of connection requests that can be pending. Connection requests will be rejected if this limit is exceeded.

The default number of pending requests is 1000. To use the default, specify -1.

## MaxConnectionThreads

Located in the `RTMP (Connector)` container.

### Description

This tag specifies the maximum number of threads used to process connection requests. The default number is 5. To use the default, specify 0.

### See also

`MinConnectionThreads`

## MaxIOThreads

Located in the ACCP, Admin, Core, ECCP, Edge, HTTP, and RTMP (Connector) containers.

### Description

This tag specifies the maximum number of threads that can be created for I/O processing.

Use the following information to configure all I/O and connection threads processing:

- A value of 0 allocates the default number of threads (10).
- A value greater than 0 allocates the exact number of threads specified.
- A value less than 0 ties the number of connection threads to the number (N) of processors, as follows:
  - -1 means 1 x N threads.
  - -2 means 2 x N threads, and so on.

Flash Media Server can receive connections on various protocols. The default value for this tag varies according to which container protocol it is nested within.

| Container | Default Value | Description |
|---|---|---|
| ACCP | 10 | Use 0 for the default value. |
| Admin | 10 | Use 0 for the default value. |
| Core | 10 | Use 0 for the default value. |
| ECCP | 10 | Use 0 for the default value. |
| Edge | 10 | Use 0 for the default value. |
| HTTP | 10 | Use 0 for the default value. |
| RTMP | 32 | Use -1 for the default value. |

### See also

MinIOThreads

## MaxQueueSize

Located in the EdgeCore, GlobalQueue, Services containers.

### Description

This tag specifies the maximum size of the queue in messages. The value is specified in kilobytes. The default size is 100K.

## MaxUnitSize

Located in the `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

### Description

This tag specifies the threshold of the maximum message size to get back into the cache. The size is specified in kilobytes. The default size is 16K.

## MaxWaitTime

### Description

This tag defines the maximum time in milliseconds that the client should wait for additional FPAD responses from the proxy servers.

### Description

The client uses the maximum wait time specified by the first valid FPAD response it receives. By default, the maximum wait time is 100 milliseconds.

### Example

```
<MaxWaitTime>100</MaxWaitTime>
```

## MessageCache

Container tag.

The tags nested within the `MessageCache` container control how the message cache holds onto messages used by the system running Flash Media Server, and keeps them in memory for reuse instead of returning them and requesting them from the operating system.

Messages are the essential communication units of Flash Media Server. Recycling them improves the server's performance.

### Contained tags

`FreeMemRatio`, `FreeRatio`, `GlobalRatio`, `MaxAge`, `MaxCacheSize`, `MaxUnitSize`, `UpdateInterval`

### See also

`LargeMemPool`, `SegmentsPool`, and `SmallMemPool` containers

## MinConnectionThreads

Located in the `RTMP (Protocol)` container.

### Description

This tag specifies the minimum number of threads in the pool for I/O operations. The default is 1 times the number of processors. To use the default, specify the value 0.

### See also

MaxConnectionThreads

## MinIOThreads

This tag is located in the ACCP, Admin, Core, ECCP, Edge, HTTP, and RTMP (Connector) containers.

The tag specifies the minimum number of threads that can be created for I/O operations.

### Description

Flash Media Server can receive connections on various protocols. The default value for this tag varies according to which container protocol it is nested within.

| Container | Default Value | Description |
|---|---|---|
| ACCP | 2X number of processors | Use 0 for the default value. |
| Admin | 2X number of processors | Use 0 for the default value. |
| Core | 2X number of processors | Use 0 for the default value. |
| ECCP | 2X number of processors | Use 0 for the default value. |
| Edge | 2X number of processors | Use 0 for the default value. |
| HTTP | 1X number of processors | Use 0 for the default value. |
| RTMP | 2X number of processors | Use -1 for the default value. |

### See also

MaxIOThreads

## MsgPoolGC

This tag specifies how often Flash Media Server checks for and removes content in the global message pool.

### Description

The default interval for checking and removing content is 60 seconds.

## MyZone

This tag specifies the zone that the edge or proxy server belongs to when it broadcasts a FPAD message. The message includes the cluster ID that the proxy or edge server belongs to.

### Description

Other edge servers in the cluster will add this edge when its zone is one of their allowed zones. Proxies respond only to other proxies with the same cluster ID.

If this tag is empty, it is assumed that the server belongs to cluster 0.

The `AllowZones` tag pertains only to client applications playing through Flash Player.

```
<MyZone>2</MyZone>
```

### See also

`AllowZones`

## NumCRThreads

Located in the `RTMP (Connector)` container.

### Description

This tag specifies the number of completion routine threads on Windows 32-bit systems for edge server I/O processing.

## Order

This tag specifies the order for evaluating the `Allow` and `Deny` tags.

### Description

This tag specifies whether Flash Media Server evaluates the `Allow` or `Deny` tag first, and the syntax it uses when evaluating these tags.

- The default sequence is `<Order>Allow,Deny</Order>`.
- If both the `Allow` and `Deny` tags are left unspecified, Flash Media Server processes all requests.
- The `<Order>Allow,Deny</Order>` sequence specifies that Flash Media Server will process the automatic proxy discovery message if the client does match something listed in the `Allow` tag and does not match anything in the `Deny` tag.
- The `<Order>Deny,Allow</Order>` sequence specifies that the automatic proxy discovery message will be processed if the client does not match anything listed in the `Deny` tag, or does match something in the `Allow` tag.

### Examples

```
<Allow>foo.macromedia.com,10.41.1.55</Allow>
<Deny>all</Deny>
<Order>Deny,Allow</Order>
```

This example instructs Flash Media Server not to process any requests except for those from the computer named foo.macromedia.com and the computer with the IP address 10.41.1.55:

```
<Allow>all</Allow>
<Deny>10.41</Deny>
<Order>Allow,Deny</Order>
```

This example specifies that server will accept and process all requests except those coming from computers that match the IP address 10.41.x.x:

### See also

`Allow`, `Deny` tags

## Process

Container tag.

### Description

The tags nested within the `Process` container contain the ID tags for all server processes. These tags are applicable for Flash Media Server running on Linux systems only.

### Contained tags

`GID`, `UID`

## Protocol

Container tag.

### Description

Flash Media Server receives connections on various protocols. The tags in this container configure those protocols and how the connection requests are received.

To set the values for all I/O and connection threads processing, follow these guidelines:

- A value of 0 allocates the default number of threads (10).
- A value greater than 0 allocates the exact number of threads specified.
- A value less than 0 ties the number connection threads to the number (N) of processors:
  - -1 means 1 x N threads
  - -2 means 2 x N threads, etc.

### Contained tags

ACCP, ECCP, RTMP (Protocol) containers

## ProxyInfo

This tag specifies the host (or IP) and port to return to the client in the response to a FPAD message.

### Description

The value for this tag must match the value for one of the HostPort tags in the Adaptor.xml file. The server must be listening on this IP address and port for the client to be able to connect to it.

If the tag is undefined when the Flash Proxy Auto-Discovery process is enabled, a warning is written to the system log. The IP address of this computer and port 1935 is returned to the clients.

### See also

HostPort in Adaptor.xml file

## ResourceLimits

Container tag.

### Description

The tags nested within the ResourceLimits container specify the maximum resource limits for the server, including the HTTP and RTMP protocols.

### Contained tags

Connector container and the ApplicationGC, CPUMonitor, FLVCacheSize, MsgPoolGC, SocketGC, SSLSessionCacheGC, ThreadPoolGC tags

### See also

Adaptor.xml file

## Root

Container tag.

### Description

The Root tag is a container for all the other tags in the Server.xml file.

## RTMP (Connector)

Flash Media Server uses two container tags named `RTMP`: one nested within the `Connector` container, and the other nested within the `Protocol` container.

Container tag located in the `Connector` container.

### Description

This container holds the tags that configure RTMP (Real-Time Messaging Protocol). RTMP is the protocol used for communication between users (typically Flash Player users) and Flash Media Server.

The following reference table lists the default values for all thread configurations.

| Default Value | Description |
|---|---|
| 0 | Allocates the default number of threads (10). |
| >0 | Allocates the exact number of threads specified. |
| <0 | Associates the default value with the number (N) of processors. |
| -1 | Allocates 1xN threads. |
| -2 | Allocates 2xN threads. |

### Contained tags

`MaxConnectionThreads`, `MaxConnectionQueueSize`, `MaxIOThreads`, `MinIOThreads`, `NumCRThreads`

### See also

`RTMP (Protocol)` in `Protocol` container.

## RTMP (Protocol)

Flash Media Server uses two container tags named `RTMP`: one nested within the `Connector` container, and the other nested within the `Protocol` container.

### Description

This container holds the tags that configure RTMP (Real-Time Messaging Protocol). RTMP is the protocol used for communication between users (typically Flash Player users) and Flash Media Server.

### Contained tags

`Admin`, `ACCP`, `Core`, `ECCP`, `Edge` containers

### See also

RTMP (Connector) in Connector container.

## Scope

This tag determines whether to write a separate log file for each virtual host or to write one log file for the server.

### Description

The value for this tag is server or vhost. The default is server, which enables logging for all processes on the server.

## SecureProxyInfo

This tag specifies the host (or IP address) and port number to return to the client in the FPAD response for clients wishing to make a secure connection.

Syntax

```
<SecureProxyInfo>[hostname/IP]:[port]</SecureProxyInfo>
```

## SegmentsPool

Container tag.

### Description

The tags in this section configure how the segments pool caches segments of FLV (Flash Video) files within Flash Media Server to increase performance of FLV streaming and keep frequently used FLV files in memory.

### Contained tags

FreeMemRatio, FreeRatio, GlobalRatio, MaxAge, MaxCacheSize, MaxUnitSize, UpdateInterval

### See also

The LargeMemPool, MessageCache, and SmallMemPool containers.

## Server

Container tag.

### Description

The tags next within the Server tag contains the tags that configure the server.

### Contained tags

AdminServer, AutoDiscovery, Logging, Mask, Process, ResourceLimits, and SSL containers

## ServerDomain

This tag specifies the host name (with the domain) of the server computer.

### Description

You set this tag in the referrer header tag when a connection is established with a remote server using NetConnection. Set this tag to the server's domain name so that it can pass the domain name to any application servers it connects to. For security purposes, some application servers require this information as a part of incoming connection requests.

If this tag is not set, the host name field is not supplied in the referrer header.

## Services

Container tag.

### Description

The tags in this section control the IPC message queue used by the edge and core processes to communicate with each other.

### Contained tags

HeapSize, MaxQueueSize

## SmallMemPool

Container tag.

### Description

The tags in this section configure the small memory pool, which saves small chunks of memory within Flash Media Server to increase performance of small allocations.

### Contained tags

FreeMemRatio, FreeRatio, GlobalRatio, MaxAge, MaxCacheSize, MaxUnitSize, UpdateInterval

### See also

The LargeMemPool, MessageCache, and SegmentsPool containers.

## SocketGC

### Description

This tag specifies in seconds how often Flash Media Server checks for and removes inactive sockets.

The default value is 60 seconds.

Located in the `AdminServer` and `ResourceLimits` containers.

## SocketOverflowBuckets

This tag specifies the number of overflow buckets if all slots in the socket table are in use.

### Description

The default number of buckets is 16. Use -1 for the default value.

Located in the `ACCP`, `Admin`, `Core`, `ECCP` containers, and in the `RTMP (Protocol)` container within the `Protocol` container.

### See also

`SocketTableSize`

## SocketTableSize

This tag specifies the size of the direct-access socket table for quick lookup.

### Description

The default size is 200. Use -1 for the default value.

Located in the `ACCP`, `Admin`, `Core`, `ECCP` containers, and in the `RTMP (Protocol)` container within the `Protocol` container.

### See also

`SocketOverflowBuckets`

## SSL

Container tag.

### Description

The SSL tags in Server.xml configure Flash Media Server to act as an SSL-enabled client by securing the outgoing connections.

The following is a quick-start to enable SSL connections with Flash Media Server.

- Specify the location of the certificate in the `SSLCertificateFile` tag.
- If the private key file is encrypted, specify the passphrase to use for decrypting the private key file in the `SSLPassPhrase` tag.
- Save the modified Server.xml file.

### Contained tags

`SSLClientCtx` container and the `SSLRandomSeed`, `SSLRandomSeed`, and `SSLSessionCacheGC` tags.

### See also

`SSLClientCtx`

## SSLCACertificateFile

This tag specifies the name of one or more digital certificates that Flash Media Server uses for SSL-based secured communications.

### Description

This tag specifies the name of a file that contains one or more CA (Certificate Authority) digital certificates in PEM (privacy enhanced mail) encryption format.

### See also

`SSLCACertificatePath`

## SSLCACertificatePath

This tag specifies the directory containing one or more CA certificates.

### Description

This tag specifies the directory containing CA certificates. Note that each file in the directory can contain only a single CA certificate, and the files must be named by the subject name's hash, and an extension of .0.

The following information is for Windows systems only:

Because Microsoft Windows installs certificates in the registry, there is no file system directory that contains all the trusted root certificates. You must import the certificates previously installed in the Windows certificate store into individual certificates and place them in a directory accessible by OpenSSL.

To import these certificates, run `FMSmaster > Console > Initialize [directory]`. This action imports all current certificates into a `certs` directory in the Flash Media Server installation directory.

When verifying a certificate, Flash Media Server will look for trusted root certificates in the file specified by the `SSLCACertificateFile` tag or in the directory specified by the `SSLCACertificatePath` tag.

If the `SSLCACertificatePath` tag is empty, Flash Media Server tries to find the root certificate in the default `certs` directory.

### See also

`SSLCACertificateFile`

## SSLCipherSuite

This tag specifies the suite of encryption ciphers that Flash Media Server uses to secure communications.

### Description

This tag is a colon-delimited list of encryption resources, such as a key exchange algorithm, authentication method, encryption method, digest type, or one of a selected number of aliases for common groupings. Each item in the cipher list specifies the inclusion or exclusion of an algorithm or cipher. In addition, there are special keywords and prefixes. For example, the keyword `"ALL"` specifies all ciphers, and the prefix `"!"` removes the cipher from the list.

The default cipher string is:

```
<SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH</SSLCipherSuite>
```

The default cipher list instructs Flash Media Server to accept all ciphers, but block those using anonymous Diffie-Hellman authentication, block low-strength ciphers, block export ciphers, block MD5 hashing, and sort ciphers by strength from highest to lowest level of encryption.

| NOTE | Contact Flash Media Server Technical Support before changing the default settings. |
|---|---|

The cipher list consists of one or more cipher strings separated by colons. Commas or spaces are also acceptable separators but colons are normally used.

The string of ciphers string can take several different forms.

- It can consist of a single cipher suite such as RC4-SHA.
- It can represent a list of cipher suites containing a certain algorithm, or cipher suites of a certain type.

  For example, `SHA1` represents all ciphers suites using the digest algorithm SHA1, and `SSLv3` represents all SSL v3 algorithms.
- Lists of cipher suites can be combined in a single cipher string using the + character as a logical `and` operation.

  For example `SHA1+DES` represents all cipher suites containing the `SHA1` and the `DES` algorithms.
- Each cipher string can be optionally preceded by the characters!, - or +.
- If ! is used then the ciphers are permanently deleted from the list. The ciphers deleted can never reappear in the list even if they are explicitly stated.
- If - is used then the ciphers are deleted from the list, but some or all of the ciphers can be added again by later options.
- If + is used then the ciphers are moved to the end of the list. This option doesn't add any new ciphers—it just moves matching existing ones.
- If none of these characters is present then the string is just interpreted as a list of ciphers to be appended to the current preference list. I
- If the list includes any ciphers already present, Flash Media Server does not evaluate them.
- The cipher string `@STRENGTH` will sort the current cipher list in order of the length of the encryption algorithm key.

The components can be combined with the appropriate prefixes to create a list of ciphers including only those ciphers Flash Media Server is prepared to accept, in the order of preference.

### Examples

```
<SSLCipherSuite>ALL:!ADH:!EDH</SSLCipherSuite>
```

This cipher string instructs Flash Media Server to accept all ciphers except those using anonymous or ephemeral Diffie-Hellman key exchange.

```
<SSLCipherSuite>RSA:!NULL!EXP</SSLCipherSuite>
<SSLCipherSuite>RSA:LOW:MEDIUM:HIGH</SSLCipherSuite>
```

These cipher strings instruct Flash Media Server to accept only RSA key exchange, and refuse export or null encryption. The server evaluates both strings as equivalent.

```
<SSLCipherSuite>ALL:+HIGH:+MEDIUM:+LOW:+EXP:+NULL</SSLCipherSuite>
```

This cipher list instructs the server to accept all ciphers, but order them in order of decreasing strength. This sequencing allows clients to negotiate for the strongest cipher that both they and the server can accept.

```
<SSLCipherSuite>ALL:+HIGH:!LOW:!EXP:!NULL</SSLCipherSuite>
```

This string instructs the server to accept only high- and medium-strength encryption, with the high being preferred, and reject export-strength versions.

```
<SSLCipherSuite>ALL:+SSLv2</SSLCipherSuite>
```

This string instructs the server to accept all ciphers, but order them so that SSLv2 ciphers come after SSLv3 ciphers:

Here is the complete list of components that Flash Media Server can evaluate:

| Key Exchange Algorithm | Description |
|---|---|
| kRSA | Key exchange |
| kDHr | Diffie-Hellman key exchange with RSA key |
| kDHd | Diffie-Hellman key exchange with DSA key |
| RSA | Ephemeral Diffie-Hellman key exchange |
| DH | RSA key exchange |
| EDH | Ephemeral Diffie-Hellman key exchange |
| ADH | Anonymous Diffie-Hellman key exchange |

| Authentication Methods | Description |
|---|---|
| aNULL | No authentication |
| aRSA | RSA authentication |
| aDSS | DSS authentication |
| aDH | Diffie-Hellman authentication |

| Encryption Methods | Description |
|---|---|
| eNULL | No encoding |
| DES | DES encoding |
| 3DES | Triple-DES encoding |
| RC4 | RC4 encoding |
| RC2 | RC2 encoding |
| IDEA | IDEA encoding |
| NULL | No encryption |
| EXP | All export ciphers (40 bit encryption) |
| LOW | Low-strength ciphers (no export, DES) |
| MEDIUM | 128-bit encryption |
| HIGH | Triple-DES encoding |

| Digest Types | Description |
|---|---|
| MD5 | MD5 hash function |

| Digest Types | Description |
|---|---|
| SHA1 | SHA1 hash function |
| SHA | SHA hash function |

| Additional Aliases | Description |
|---|---|
| All | All ciphers |
| SSLv2 | All SSL version 2.0 ciphers |
| SSLv3 | All SSL version 3.0 ciphers |
| DSS | All ciphers using DSS authentication |

## SSLClientCtx

Container tag.

### Description

The tags in this section configure Flash Media Server to perform as an SSL client for outgoing connections.

### Contained tags

`SSLVerifyCertificate`, `SSLCACertificateFile`, `SSLCACertificatePath`, `SSLVerifyDepth`, `SSLCipherSuite`

### See also

The `SSL` container.

## SSLRandomSeed

This tag specifies the number of bytes of entropy to use for seeding the pseudo-random number generator (PRNG).

### Description

Entropy is a measure of randomness. The more entropy, the more random are the numbers that the PRNG will generate. The default number of bytes to specify for this tag is 16. Specifying a larger number for this tag provides improved randomness and therefore better security, but the larger number may noticeably affect the server's performance. If security is a primary concern for your applications, you should experiment to determine the best value for this tag.

| NOTE | You cannot specify less than 8 bytes. |
|---|---|

### See also

`SSLCipherSuite`, `SSLRandomSeed`

## SSLSessionCacheGC

This tag specifies in minutes how often to check for and remove unused server-to-server connections from the SSL session cache.

### Description

When Flash Media Server establishes a secure connection to another server, it automatically caches the session ID (when session caching is supported by the server being connecting to) to avoid performing full SSL handshakes with this server in the future.

The default interval is 5 minutes.

## SSLVerifyCertificate

This tag instructs Flash Media Server whether or not to verify the certificate returned by the server being connected to.

### Description

Certificate verification is enabled by default. To disable certificate verification, set the value for this tag to "`false`".

```
<SSLVerifyCertificate>false</SSLVerifyCertificate>
```

| WARNING | Disabling certificate verification can result in a security hazard. |
|---|---|

### See also

SSLVerifyDepth

## SSLVerifyDepth

This tag specifies the maximum depth in the certificate chain from which Flash Media Server will accept certificates.

### Description

If a self-signed root certificate cannot be found within the specified depth, the certification verification will fail. The default depth is 9.

### See also

SSLVerifyCertificate

## ThreadPoolGC

This tag specifies in minutes how often Flash Media Server checks for and removes unused I/O threads.

### Description

The default time is 20 minutes. You cannot specify less than 20 minutes.

## Time

This tag specifies the time field in a log file.

### Description

The time field in a log file can be specified either as UMT (GMT) or local time. The default setting is local.

## TTL

This tag specifies in seconds how often to broadcast a keep-alive message to other edges in the cluster, and how often another edge should expect to receive a keep-alive message from this edge.

The default value is one second.

### Description

If the other edges do not receive the keep-alive message within the specified `TTL` limit, the FADP assumes that this edge server is not operating and removes it from the cluster.

### See also

ClusterMonitorInterval

## UID

This tag contains the server process user ID.

### Description

If no UID or group ID (GID) is specified, the server will run as root. This tag is applicable for Flash Media Server running on Linux systems only.

### See also

GID

## UpdateInterval

### Description

This tag specifies how often, per reused messages, thread statistics are collected.

### Description

The default count is every 1024 messages.

Located in the LargeMemPool, MessageCache, SegmentsPool, and SmallMemPool containers.

## UserData

This tag specifies the user data that is returned to the client via the FPAD response.

### Description

Each subtag represents a user-defined property of NetConnection.proxyInfo.

### Example

```
<UserData>
  <foo>bar</foo>
</UserData>
```

This result for this subtag displays the following property:

```
NetConnection.proxyInfo.foo = "bar".
```

# Users.xml file

Users.xml is the configuration file for the Flash Media Admin Service users and is located at the root level of the `conf` directory. It contains the tags and information used to identify the Flash Media Server administrators and their access permissions. You edit the Users.xml file to add or remove Flash Media Server administrators, or change their administrative permissions.

The Users.xml file contains the following tag structure.

```
<Root>
  <UserList>
    <User name="${SERVER.ADMIN_USERNAME}">
      <Password encrypt=false>${SERVER.ADMIN_PASSWORD}</Password>
      <Allow></Allow>
      <Deny></Deny>
      <Order>Allow,Deny</Order>
    </User>
  </UserList>
  <AdminServer>
       <HTTPCommands>
         <Enable></Enable>
         <Allow></Allow>
         <Deny></Deny>
         <Order></Order>
       </HTTPCommands>
  </AdminServer>
</Root>
```

# Summary of Users.xml tags

This table lists alphabetically the tags in the Users.xml configuration file.

| Users.xml tag | Description |
|---|---|
| `AdminServer` | Container tag; contains tags to configure access to the Flash Media Admin Service. |
| `Allow (HTTPCommands)` | Defines the list of server administrative commands that can be accessed via HTTP. |
| `Allow (Users)` | Defines the list of specific hosts from which the administrator can connect to Flash Media Admin Service. |
| `Deny (HTTPCommands)` | Lists the Flash Media Admin Service commands denied access via HTTP. |
| `Deny (User)` | Lists the specific hosts from which the administrator cannot connect to the Flash Media Admin Service. |

| Users.xml tag | Description |
|---|---|
| Enable | Enables or disables using HTTP requests to execute admin commands. |
| HTTPCommands | Container tag; contains settings for those admin commands accessed through the HTTP protocol. |
| Order (HTTPCommands) | Specifies the order of processing for lists of denied and allowed HTTP commands for accessing the Flash Media Admin Service. |
| Order (User) | Specifies the order in which to evaluate the Allow and Deny tags. |
| Password | Specifies the password for this virtual host administrator. |
| Root | Root tag; this tag is a container for all the other tags. |
| User | Identifies an administrator of the server. |
| UserList | Container tag; defines the access permissions for administrators of the Flash Media Admin Service. |

# Description of Users.xml tags

The following alphabetical list of Users.xml tags contains additional information, including cross references to associated tags, syntax, and examples.

## AdminServer

Container tag

### Description

The HttpCommands container nested within the AdminServer container configures the access level to the Flash Media Admin Service.

The Admin Service is separate from Flash Media Server. When administrators use the management console to connect to Flash Media Server, they are connecting to the Flash Media Admin Service, which in turn connects to the server.

### Contained tag

HTTPCommands container

## Allow (HTTPCommands)

Flash Media Server uses two tags named Allow: the Allow tag in the User container, and the Allow tag in the HTTPCommands container.

### Description

This tag lists the Flash Media Admin Service commands that the administrator can access using HTTP. You can authorize an administrator to use multiple HTTP commands for access by creating a comma-separated list of the commands.

Using the value "All" for the `Allow` tag authorizes the administrator to use all HTTP commands.

| NOTE | Macromedia does not recommend use of "all," as it creates an opportunity for a security risk. |
|---|---|

### See also

`Deny (HTTPCommands)` in the HTTPCommands container; `Allow (Users)` in the User container.

## Allow (Users)

Flash Media Server uses two tags named `Allow`: the `Allow` tag in the `User` container, and the `Allow` tag in the `HTTPCommands` container.

### Description

This tag lists the specific hosts from which an administrator can connect to the Flash Media Admin Service. The administrator can only connect to the server from those hosts specified in this `Allow` tag.

You authorize the administrator's access by creating a comma-delimited list of the accessible host names or domain names, and/or full or partial IP addresses. Whenever possible, use the IP addresses in the `Allow` tag to improve the server's performance when processing connection requests.

### Example

```
<Allow>foo.yourcompany.com,macromedia.com,10.60.1.133,10.60</Allow>
```

### See also

`Deny (User)` in the `User` container; `Allow (HTTPCommands)` in the `HTTPCommands` container.

## Deny (HTTPCommands)

Flash Media Server uses two tags named `Deny`: the `Deny` tag in the `User` container, and the `Deny` tag in the `HTTPCommands` container.

### Description

This `Deny` tag lists the Flash Media Admin Service commands that an administrator cannot use via HTTP.

You can deny an administrator the use of multiple HTTP commands to access the Admin Service by creating a comma-separated list of those HTTP commands.

### See also

`Allow (HTTPCommands)` in the HTTPCommands container; `Deny (User)` in the User container.

## Deny (User)

Flash Media Server uses two tags named `Deny`: the `Deny` tag in the `User` container, and the `Deny` tag in the `HTTPCommands` container.

This tag lists those hosts from which the administrator is not authorized to connect to Flash Media Admin Service. You restrict the administrator's access by creating a comma-delimited list of those host names or domain names and/or (full or partial) IP addresses.

### Example

```
<Deny>foo.yourcompany.com,macromedia.com,10.60.1.133,10.60</Deny>
```

This example lists the computers sending connections requests that Flash Media Admin Service will not accept.

### See also

`Allow (Users)` in the `User` container; `Deny (HTTPCommands)` in the HTTPCommands container.

## Enable

This tag enables or disables the use of HTTP requests to execute administrative commands.

### Description

Setting this tag enables HTTP requests to execute administrative commands. To disable administrative access through the use of HTTP requests, do not set this tag.

### Syntax

```
<Enable>on</Enable>
```

or

```
<Enable></Enable>
```

## HTTPCommands

Container tag.

### Description

This section contains the settings for those Flash Media Admin Service commands that can be accessed through HTTP.

### Contained tags

`Allow (HTTPCommands)`, `Deny (HTTPCommands)`, `Enable`, `Order (HTTPCommands)`

## Order (HTTPCommands)

Flash Media Server uses two `Order` tags: one in the `HTTPCommands` container, and the other in the `User` container.

### Description

This tag specifies the order for evaluating the `Deny` and `Allow` commands.

### Syntax

```
<Order>Deny,Allow</Order>
```

The sequence `Deny, Allow` means the HTTP command will be allowed if the command is in the `Allow` list of commands or not in the `Deny` list.

```
<Order>Allow,Deny</Order>
```

The sequence `Allow, Deny` means the HTTP command will be allowed if it is in the `Allow` list of commands and not in the `Deny` list:

### See also

`Allow (HTTPCommands)`, `Deny (HTTPCommands)`

## Order (User)

Flash Media Server uses two `Order` tags: one in the `HTTPCommands` container, and the other in the `User` container.

### Description

This tag specifies the sequence in which Flash Media Server evaluates the `Allow` and `Deny` tags for an administrator.

### Syntax

```
<Order>Allow,Deny</Order>
```

The default sequence `Allow, Deny` means that administrative access is allowed unless the user is specified in the `Allow` list of commands and not in the `Deny` list:

```
<Order>Deny,Allow</Order>
```

The alternative sequence `Deny, Allow` means that administrative access is allowed unless the user is specified in the `Deny` list of commands and not specified in the `Allow` list:

### See also

`Allow (Users)`, `Deny (User)`

## Password

This tag specifies the password for the administrator of this vhost.

### Description

Passwords cannot be empty strings (“”). Passwords are usually encrypted. In the following example, the `encrypt` attribute instructs the server to encrypt the contents of the password. When the `encrypt` attribute is set to `true`, the password you see in the file is the encrypted password, and it is interpreted as an encoded string.

### Example

```
<Password encryt=”true”></password>
```

## Root

Container tag.

### Description

The `Root` tag is a container for all the other tags.

## User

This tag identifies an administrator for the server.

### Description

You can identify multiple administrators for a virtual host by creating a profile for each administrator with the `User`, `Password`, `Allow (Users)`, `Deny (User)`, and `Order (User)` tags.

### Example

```
<User name=”jsmith></User>
```

Use the `name` attribute to identify the login name of a Flash Media Server administrator:

## UserList

Container tag.

### Description

The `UserList` tag defines the access permissions for administrators that use the Flash Media Admin Service.

### Description

Each administrator is defined with the `User`, `Password`, `Allow (Users)`, `Deny (User)`, and `Order (User)` tags.

# Logger.xml file

The Logger.xml file is located at the root level of the `conf` directory and is the configuration file for the logging file system. Logger.xml contains the tags and information used to configure the Flash Media Server log files. You can edit this file to add or change configuration information, including the location of the log files. The default location of the log files is in the `logs` directory in the server installation directory.

Log files are written in English. Field names displayed in the log file are in English. Some content within the log file, however, may be in another language, depending on the filename and the operating system. For example, in the Access.log file, the columns x-sname and x-suri-stem show the name of the stream. If the name of the recorded stream is in a language other than English, the stream's name will be written in that language, even if the server is running on an English-language operating system.

The `Logging` section in the Server.xml enables or disables the log files. Tags to configure the log files are in the Logger.xml file.

The Logger.xml file contains the following tag structure.

```
<Logger>
  <Access>
    <LogServer enable="false" type="udp"></LogServer>
      <HostPort></HostPort>
      <ServerID></ServerID>
      <DisplayFieldsHeader>100</DisplayFieldsHeader>
    <Directory>${LOGGER.LOGDIR}</Directory>
    <FileName>access.[NN].log</FileName>
    <Time>local</Time>
    <Rotation>
      <MaxSize>10240</MaxSize>
      <Schedule type="daily">00:00</Schedule>
      <History>5</History>
    </Rotation>
```

```
    <Events>connect;disconnect;play;pause;unpause;stop</Events>
    <Fields>x-category;x-event;date;time;x-pid;c-ip;cs-bytes;sc-bytes;
  x-sname;sc-stream-bytes;x-file-size;x-file-length</Fields>
    <Delimiter></Delimiter>
    <QuoteFields>disable</QuoteFields>
    <EscapeFields>enable</EscapeFields>
  </Access>
  <Diagnostic>
    <Directory>${LOGGER.LOGDIR}</Directory>
    <Rotation>
      <MaxSize>10240</MaxSize>
      <Schedule type="daily">00:00</Schedule>
      <History>5</History>
    </Rotation>
  </Diagnostic>
  <Application>
    <Directory>${LOGGER.LOGDIR}</Directory>
    <Rotation>
      <MaxSize>10240</MaxSize>
      <Schedule type="daily">00:00</Schedule>
      <History></History>
    </Rotation>
  </Application>
</Logger>
```

NOTE

Log file rotation cannot be disabled. To effectively turn off rotation, choose a large maximum size and a long maximum duration for the log files.

## Summary of Logger.xml tags

This table lists alphabetically the tags in the Flash Media Server Logger.xml configuration file. By default, the log files are located in the logs directory in the server installation directory.

| Logger.xml tag | Description |
|---|---|
| Access | Container tag; contains tags to configure the Access log file settings. |
| Application | Container tag; contains tags to configure the Application log file settings. |
| Delimiter | Specifies which delimiter to use when separating the fields in the log file. |
| Diagnostic | Container tag; contains tags to configure the diagnostic log file settings. |
| Directory | Specifies how many lines to write to log file before repeating the field headers. |

| Logger.xml tag | Description |
|---|---|
| EscapeFields | Formatting tag; specifies whether or not unsafe characters in the log file are escaped. |
| Events | Specifies the events written to the Access log file. |
| Fields | Specifies which fields for an event are logged in the Access log file. |
| FileName | Specifies the name of the log files. |
| History | Specifies the maximum number of log files to keep. |
| HostPort | Specifies the IP and port number of the log server. |
| Logger | Root tag; this tag is a container for all the other tags. |
| LogServer | Container tag; contains tags to configure the server to send messages to a remote log server. |
| MaxSize | Specifies the maximum size of the log files. |
| QuoteFields | Formatting tag; specifies whether or not to use quotation marks to surround those fields in the log file that include a space. |
| Rotation | Container tag; contains tags to configure the rotation of the log files. |
| Schedule | Specifies how frequently the log files are rotated. |
| ServerID | Identifies by IP address the Flash Media Server whose logged events are being recorded. |
| Time | Specifies the time zone for a log file. |

# Description of Logger.xml tags

The following alphabetical list of Logger.xml tags contains additional information, including cross references to associated tags, syntax, and examples.

## Access

Container tag.

### Description

The tags nested within this container configure the Access log settings.

### Contained tags

Delimiter, EscapeFields, Events, Fields, FileName, LogServer, QuoteFields, Rotation, Time

## Application

Container tag.

### Description

The tags nested within this container configure the Application log file settings.

### Contained tags

Directory, Rotation, Time

## Delimiter

Formatting tag. This tag specifies whether or not to use single quotation marks (‘ ‘) as a delimiter to separate the fields in the log file.

### Description

A delimiter is used to separate the fields in the log file. The use of the pound sign (#) as a delimiter is not recommended, since # is used as the comment tag in the Logger.xml file.

The following characters are not allowed as delimiters:

- triple quotation marks (‘”)
- paired double quotation marks (““)
- comma (,)
- colon (:)
- hyphen (-)

### See also

Directory, EscapeFields, QuoteFields

## Diagnostic

Container tag.

### Description

The tags in this section configure the diagnostic log file.

### Contained tags

`Directory`, `Rotation`. `Time`

## Directory

This tag specifies the directory where the log files are located.

### Description

By default, the log files are located in the logs directory in the server installation directory.

Located in `Access`, `Application`, `Diagnostic` containers

## DisplayFieldsHeader

Formatting tag. This tag specifies how many lines to write to the log file before repeating the field headers.

### Description

The default line count is 100 lines.

### See also

`Delimiter`, `EscapeFields`, `QuoteFields`

## EscapeFields

Formatting tag. This tag controls whether or not the fields in the log file are escaped when unsafe characters are found.

### Description

This optional flag can be set to `enable` or `disable`. By default, it is set to `enable`.

The unsafe characters are as follows: the space character; open or closed angle brackets (< >); a double quotation mark ("); the pound sign (#); the percent sign (%); open or closed curly braces ({ }); bars (|); carat (^); tilde (~); square brackets ([ ]); and apostrophe (`).

### See also

`Delimiter`, `Directory`, `QuoteFields`

## Events

Events are written to the log file.

### Description

The following table lists the events recorded in the Access log file. Events are logged in a semicolon-separated list. The keyword * instructs Flash Media Server to log all events.

| Event | Category | Description |
|---|---|---|
| app-start | application | Application instance starts. |
| app-stop | application | Application instance stops. |
| connect | application | Client connects to the server. |
| connect-pending | application | Client connects to the server, waiting for the script to authenticate. |
| disconnect | application | Client disconnects. |
| pause | application | Client pauses a recorded stream. |
| play | application | Client plays a recorded or live stream. |
| publish | application | Client publishes a live stream. |
| record | application | Client begins recording a stream. |
| recordstop | application | Client stops recording a stream. |
| seek | application | Client jumps to a new location within a recorded stream. |
| server-start | application | Server has started. |
| server-stop | application | Server has stopped. |
| stop | application | Client stops playing a recorded or live stream or stops publishing a live stream. |
| unpause | application | Client resumes a stream. |
| unpublish | application | Client unpublishes a live stream. |
| vhost-start | application | A virtual host has started. |
| vhost-stop | application | A virtual host has stopped. |

The following events display a status code.

| Field | Status Code | Description |
|---|---|---|
| connect-pending | 100 | Waiting for the application to authenticate. |
| connect | 200 | Successful connection. |
| | 302 | Application currently unavailable. |

| Field | Status Code | Description |
|---|---|---|
| | 400 | Bad request; client connected to server using an unknown protocol. |
| | 401 | Connection rejected by the application script. |
| | 403 | Connection rejected by access module. |
| | 404 | Application not found. |
| | 409 | Resource limit exceeded. |
| | 413 | License limit exceeded. |
| | 500 | Server internal error. |
| | 502 | Bad gateway. |
| | 503 | Service unavailable; for instance, too many connections pending for authorization by access module. |
| play | 200 | Successful. |
| | 400 | Bad request (invalid arguments). |
| | 401 | Access denied by application. |
| | 403 | Play forbidden by stream module. |
| | 404 | Stream not found. |
| | 415 | Unsupported media type. |
| | 500 | Server internal error. |
| publish | 200 | Successful. |
| | 400 | Bad request (invalid arguments). |
| | 401 | Access denied by application. |
| | 409 | Stream is already being published |
| | 415 | Unsupported media type. |
| | 500 | Server internal error. |
| stop | 200 | Successful. |
| | 408 | Stream stopped because client disconnected. |

### See also

Fields

## Fields

This tag specifies which fields for an event are logged in the Access log file.

### Description

Fields are associated with the events found in the Access log file. The field specification is a semicolon-separated list of one or more fields associated with an event in the log file.

The keyword * specifies that all fields are to be logged. Fields without data are left empty. Macromedia recommends that you include the following fields in the fields to be logged: the type, category, date, and time fields.

The following table is a complete list of fields associated with events in the Access log file. Not every field is associated with each event in the log file.

| Field | Event(s) | Description |
|---|---|---|
| x-event | application | Type of event. |
| x-category | application | Event category. |
| date | application | Date at which the event occurred. |
| time | application | Time at which the event occurred. |
| tz | application | Time zone information. |
| x-ctx | application | Event-dependent context information. |
| x-pid | application | Server process ID. |
| x-cpu-load | application | CPU load. |
| x-mem-load | application | Memory usage (as reported by the `getServerStats()` method). |
| x-adaptor | application | Adaptor name. |
| x-vhost | application | Vhost name. |
| x-app | application | Application names. |
| x-appinst | application | Application instance names. |
| c-ip | application | Client IP address. |
| c-proto | application | Connection protocol: RTMP or RTMPT. |
| s-uri | application | URI of the Flash Media Server application. |
| c-referrer | application | URI of the referrer. |
| c-user-agent | application | User agent. |
| c-client-id | application | Client ID. |
| cs-bytes | application | This field shows the number of bytes transferred from the client to the server.<br>This information can be used to bill customers per session. To calculate the bandwidth usage per session, subtract the value of 'cs-bytes' in the 'connect' event from the value of 'cs-bytes' in the 'disconnect' event. |

| Field | Event(s) | Description |
|---|---|---|
| sc-bytes | application | This field shows the number of bytes transferred from the server to the client.<br>This information can be used to bill customers per session. To calculate the bandwidth usage per session, subtract the 'sc-bytes' in the 'connect' event by the 'sc-bytes' in the 'disconnect' event |
| x-sname | application | Stream name. |
| x-file-size | application | Stream size in bytes. |
| x-file-length | application | Stream length in seconds. |
| x-spos | application | Stream position. |
| cs-stream-bytes | application | This field shows the number of bytes transferred from the client to the server per stream.<br>To calculate the bandwidth usage per stream, subtract the 'cs-stream-bytes' in the 'publish' event by the 'cs-stream-bytes' in the 'unpublish' event. |
| sc-stream-bytes | application | This field shows the number of bytes transferred from the server to the client per stream.<br>To calculate the bandwidth usage per stream, subtract the 'sc-stream-bytes' in the 'play' event by the 'sc-stream-bytes' in the 'stop' event. |
| cs-uri-stem | application | Stem portion of s-uri (omitting query) field. |
| cs-uri-query | application | Query portion of s-uri. |
| x-sname-query | application | Query portion of stream URI specified in play or publish. |
| x-file-name | application | Full path of the file representing x-sname stream. |
| x-file-ext | application | Stream type. Currently Flash Media Server supports FLV or MP3 files. |
| s-ip | application | IP address or addresses of the server. |
| x-duration | application | Duration of a stream or session event. |
| x-suri-query | application | Same as x-sname-query. |
| x-suri-stem | application | This is a composite field: cs-uri-stem + x-sname + x-file-ext. |
| x-suri | application | This is a composite field: cs-uri-stem + x-sname + x-file-ext + x-sname-query. |
| x-status | application | See the following table for a complete description of the x-status codes and descriptions. |

### See also

Events

## FileName

This tag specifies the name of the Access log file.

### Description

The Access log file name includes a date stamp and version number. Y represents the year of its creation; the format YYYY must be used. M represents the month of its creation; the formats M or MM are both allowed. D represents the day of the month of the file's creation; the formats D or DD are both allowed. N represents the version number of the file. Note that there is no limit on number of versions.

The repetition of a letter represents the number of digits. For example, M represents 4 (April). MM represents 04 (April).

### Syntax

```
access.[YYYYMMDDNN].log
```

### Example

```
access.2005103043.log
```

This example identifies version 43 of the access log file for October 10, 2005.

## History

This tag specifies the maximum number of log files to keep.

### Description

The files are named as access.01.log, access.02.log, access.03.log, and so on. The default number of files to retain is 5.

## HostPort

This tag specifies the IP and port of the log server.

### Syntax

```
[IP]:[port]
```

### Example

```
<HostPort>xxx.xxx.xxx.xxx:1234</HostPort>
```

## Logger

Root tag.

### Description

The `Logger` tag is a container for all the other tags in Logger.xml.

## LogServer

Container tag.

### Description

The tags nested in this section configure the server to send messages to a remote log server.

### Contained tags

`Directory`, `HostPort`, `ServerID`

### See also

`ServerID`

## MaxSize

This tag specifies the maximum log file size in bytes. The default file size is 10240Kb, or approximately 1 Mb.

### Example

```
<Maxsize>10240</MaxSize>
```

### See also

`Schedule`

## QuoteFields

Formatting tag. Specifies whether or not to use quotation marks to surround those fields in the log file that include a space.

### Description

This tag can be set to `enable` or `disable`. By default, it is set to `disable`.

### See also

`Delimiter`, `EscapeFields`

## Rotation

Container tag.

### Description

The tags in this section configure the rotation of the log files.

Located in `Access`, `Application`, `Diagnostic` containers.

### Contained tags

`History`, `MaxSize`, `Schedule`

## Schedule

This tag specifies the rotation schedule for the log files.

### Description

There are two types of scheduling: daily rotation and rotation that occurs when the log exceeds a specified length.

### Examples

```
<Schedule type="daily"></Schedule>
```

If the `type` attribute is `daily`, Flash Media Server rotates the log files every 24 hours.

```
<Schedule type="hh:mm"></Schedule>
```

If the `type` attribute is `hh:mm`, the timestamp `00:00` causes the file to rotate every midnight.

```
<Schedule type="duration"></Schedule>
```

If the `type` attribute is `duration`, rotation occurs when the duration of the log exceeds a specified length. The duration is specified in minutes.

Located in `Access`, `Application`, `Diagnostic` containers

### See also

`MaxSize`,`Time`

## ServerID

By default, the value of the `ServerID` tag is the IP address of the server whose events are being logged.

### See also

`LogServer`

## Time

The `Time field` in a log file can be logged either in UTC (GMT) or local time.

### Description

The setting for the `Time` tag can be used to override the server-wide configuration. The default is local time.

### See also

The `Logging` container in the Server.xml file.

# Adaptor.xml file

The Adaptor.xml file is the configuration file for individual network adaptors in Flash Media Server. It determines the number of threads that can be used by the adaptor, the communications ports that adaptor binds to, and the IP addresses or domains from which the adaptor can accept connections.

You can also implement SSL with the Adaptor.xml file, if you want to use different digital certificates for different adaptors.

Each adaptor has its own directory inside the server's `conf` directory. The name of the directory is the name of the adaptor. Each adaptor directory must contain an Adaptor.xml file.

For example, the default adaptor included with the server at installation is named _defaultRoot_, and its directory is found in the conf/ directory. To change an adaptor's settings, you edit the tags in its Adaptor.xml file.

The Adaptor.xml file contains the following tag structure:

```
<Adaptor>
  <ResourceLimits>
    <MaxFailures></MaxFailures>
    <RecoveryTime></RecoveryTime>
  </ResourceLimits>
  <HostPortList>
    <HostPort name="xxxx"></HostPort>
  </HostPortList>
  <Allow></Allow>
  <Deny></Deny>
  <Order></Order>
  <HTTPTunnel>
    <Enable></Enable>
    <NodeID></NodeID>
    <IdlePostInterval></IdlePostInterval>
    <IdleAckInterval></IdleAckInterval>
    <MimeType></MimeType>
    <WriteBuffSize></WriteBuffSize>
```

```
    <SetCookie></SetCookie>
    <Redirect enable=”false” maxbuf=”16384”></Redirect>
    <NeedClose></NeedClose>
    <MaxWriteDelay></MaxWriteDelay>
  </HTTPTunnel>
  <SSL>
    <SSLServerCtx>
      <SSLCertificateFile></SSLCertificateFile>
      <SSLCertificateKeyFile></SSLCertificateKeyFile>
      <SSLPassPhase></SSLPassPhase>
      <SSLCipherSuite></<SSLCipherSuite>
      <SSLTimeout></SSLTimeout>
    </SSLClientCtx>
  <HTTPIdent enable="false"></HTTPIdent>
  <HTTPUserInfo enable="false"></HTTPUserInfo>
    <Path></Path>
    <MaxSize>100</MaxSize>
    <UpdateInterval>5</UpdateInterval>
</Adaptor>
```

# Summary of Adaptor. xml tags

This table lists alphabetically the tags in the Flash Media Server Adaptor.xml configuration file.

| Adaptor.xml tag | Description |
|---|---|
| `Adaptor` | Root tag; contains all the other adaptor configuration tags. |
| `Allow` | Identifies the specific hosts from which clients can connect to the server. |
| `Deny` | Identifies those hosts whose clients’ attempts to connect to the server(s) will be rejected. |
| `Enable` | Enables or disables tunneling connections into this application. |
| `HostPort` | Specifies the IP address and port(s) to bind to. |
| `HostPortList` | Contains a list of `HostPort` tags. |
| `HTTPIdent` | Configures the server to respond to or reject an HTTP identification request from a client. |
| `HTTPTunnel` | Container tag; the tags in this section configure the incoming HTTP tunneling connections. |
| `HttpUserInfo` | Specifies the physical location where the user-defined XML files are stored in the server. |

| Adaptor.xml tag | Description |
|---|---|
| IdleAckInterval | Specifies the maximum time the server may wait before it returns an ack (acknowledgement code) for a client idle post. |
| IdlePostInterval | Specifies the interval at which the client should send idle posts to the server to indicate that the player has no data to send. |
| MaxFailures | Specifies the maximum number of failures an edge server may incur before restarting. |
| MaxSize | Specifies the maximum number of XML files cached in the server. |
| MaxWriteDelay | Specifies how long the server waits for a write. |
| MimeType | Specifies the default MIME type header sent on tunnel responses. |
| NeedClose | Specifies whether HTTP 1.0 non-keepalive connections are to be closed once the response is written. |
| NodeID | Specifies a unique node identification to support the implementation of load balancers. |
| Order | Specifies the order in which to evaluate the Allow and Deny tags. |
| Path | Specifies the location of the UserInfo directory where the user-defined XML files are stored. |
| RecoveryTime | Specifies the wait time for an edge or proxy server to pause after failing before it restarts. |
| Redirect | Specifies whether or not the adaptor redirects unknown requests to an external server. |
| ResourceLimits | Container tag; contains tags that configure the resources for an edge server. |
| SetCookie | Specified whether the adaptor sets a cookie. |
| SSL | Container tag; contains tags that configure Flash Media Server to act as SSL-enabled server for secure communications. |
| SSLCACertificateFile | Specifies the name of a file that contains one or more CA certificates in the PEM encryption format. |
| SSLCACertificatePath | Specifies the name of the directory containing one or more CA certificates. |

| Adaptor.xml tag | Description |
|---|---|
| SSLCipherSuite | Specifies the encryption ciphers that Flash Media Server uses to secure incoming connections. |
| SSLClientCtx | Container tag; contains tags to configure Flash Media Server as an SSL (Secure Socket Layer) client for outgoing SSL connections. |
| SSLSessionTimeout | This tags specifies in minutes how long a SSL session remains valid. |
| UpdateInterval | Specifies how frequently the edge server checks the cache and updates the cache's contents if the XML files have changed. |
| WriteBufferSize | Specifies the size in kilobytes of the write buffer. |

# Description of Adaptor.xml tags

The following alphabetical list of Adaptor.xml tags contains additional information, including cross references to associated tags, syntax, and examples.

## Adaptor

Root tag.

### Description

The `Adaptor` tag contains all the tags in the Adaptor.xml file

## Allow

This tag identifies the specific hosts from which clients can connect to the server.

### Description

The `Allow` tag is a comma-delimited list of host names or domain names, and/or full or partial IP addresses.

### Example

```
<Allow>foo.yourcompany.com, macromedia.com, 10.60.1.133, 10.60</Allow>
```

### See also

`Deny`, `Order`

## Deny

This tag identifies the hosts whose clients' attempts to connect to the server(s) will be rejected.

### Description

The `Deny` tag is a comma-delimited list of host names or domain names and/or full or partial IP addresses.

### Example

```
<Deny>foo.yourcompany.com, macromedia.com, 10.60.1.133, 10.60</Deny>
```

### See also

`Allow`, `Order`

## Enable

This tag specifies whether or not to allow HTTP tunneling connections into the application.

### Description

The following table identifies the attributes for the `Enable` tag and describes their effect.

| Value | Description |
|---|---|
| true | Allow all HTTP tunneling connections. |
| false | Disallow all HTTP tunneling connections. |
| http1.1only | Allow only HTTP 1.1 tunneling connections. |
| keepalive | Allow HTTP 1.1 or HTTP 1.0 keepalive connections. |

**WARNING**

Although you can assign any port number for HTTP tunneling, there is a risk of conflict with another application that may be assigned to the same port. For example, if you configure Flash Media Server to use port 80 to support HTTP tunneling, the web server and Flash Media Server can not both bind to the same port 80.

## HostPort

This tag specifies which IP address and port(s) Flash Media Server is to bind to.

### Description

If you wish to bind to multiple IP addresses on this adaptor, add a `HostPort` tag for each additional IP that you wish to bind to.

### Syntax

The connection string for the `HostPort` tag has the following syntax:

```
<HostPort><ip>:<port>,<port>,...,<port></HostPort>
```

### Examples

```
<HostPort>:1935,80,-443</HostPort>
```

A port is marked as secure by specifying a minus sign in front of the port number in the `HostPort` tag. This specifies that Flash Media Server will listen on any interface, on ports 1935, 80, and 443, where 443 is designated as a secure port that will only receive RTMPS connections. Attempting an RTMPS connection to ports 1935 or 80 will result in a failure to connect. The client will attempt to perform an SSL handshake that the server will fail to complete. Similarly, a regular RTMP connection to port 443 will fail because the server will try to perform an SSL handshake that the client will fail to complete.

```
<HostPort name="edge1">127.0.0.1:1935,80,443</HostPort>
```

This `HostPort` string instructs the adaptor to bind to the IP address for Edge1 at IP address 127.0.0.1 on ports 1935, 80, and 443.

```
<HostPort>:1935,80,443</HostPort>
```

You can also bind to any IP by not specifying anything in front of the colon. This string instructs the adaptor to bind to any IP on ports 1935, 80, and 443.

```
<HostPort>127.0.0.1</HostPort>
```

If no colon is found in the `HostPort` string, the data is assumed to be an IP address and will bind to port 1935 as the default. The following string instructs the adaptor to bind to IP 127.0.0.1 on port 1935.

```
<HostPort>127.0.0.1:</HostPort>
```

When a colon is found but no ports are specified after it, port 1935 is used as the default port in which to bind. This string instructs the adaptor to bind to IP 127.0.0.1 on port 1935.

When assigning port numbers, keep in mind the following:

- There is a risk in assigning more than one adaptor to listen on the same `IP:port` pair. If another process tries to bind to the same `IP:port` combination, a conflict results.

  To resolve this conflict, the first adaptor to bind to the specified `HostPort` wins. Flash Media Server logs a warning in the Access log file indicating that the specified `IP:port` is in use.

- Although you can assign any port number for HTTP tunneling, there is a risk of conflict with another application that may be assigned to the same port.

  For example, if you configure Flash Media Server to use port 80 to support HTTP tunneling, a web server and Flash Media Server can not both bind to port 80.

### See also

HostPortList

## HostPortList

This tag contains a list of HostPort tags associated with this adaptor.

### Example

```
<HostPort secure="true">12.34.56.78:443</HostPort>
<HostPort secure="false">12.34.56.78:1935,80</HostPort>
```

This example demonstrates how to list secure and non-secure ports.

Although you can use any port number, there is a risk of conflicting with another application that may be assigned to the same port. For example, if you configure Flash Media Server to use port 80 to support HTTP tunneling, you can not bind both a web server and Flash Media Server to the same port 80.

### See also

HostPort

## HTTPIdent

This tag configures the server to respond to or reject an HTTP identification request from a client.

### Example

```
<HTTPIdent enable="true"></HTTPIdent>
```

When the enable attribute is set to "true," all tags within the HTTPIdent section are returned as a response. The entire response will be enclosed in <FCS></FCS> tags, which are added by the server.

If the HTTPIdent function is enabled but no content is specified, the <FCS></FCS> response is returned without content.

For an ident response to be returned, the HTTPIdent function must be enabled and the client must specifically do a POST or GET for "/fms/ident" resource.

### Example

```
http://localhost:1935/fms/ident
```

This command sends an HTTP get request:

```
GET /fms/ident HTTP/1.1..
Accept: */*..
Accept-Language: en-us..
```

```
Accept-Encoding: gzip, deflate..
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)..
Host: localhost:1935..
Connection: Keep
```

## HTTPTunnel

Container tag.

### Description

The tags in this section configure the incoming HTTP tunneling connections to the adaptor. Although you can assign any port number for HTTP tunneling, there is a risk of conflict with another application that may be assigned to the same port. For example, if you configure Flash Media Server to use port 80 to support HTTP tunneling, you can not bind the web server and Flash Media Server to the same port 80.

### Contained tags

`Enable`, `IdleAckInterval`, `IdlePostInterval`, `MaxWriteDelay`, `MimeType`, `NeedClose`, `NodeID`, `Redirect`, `SetCookie`, `WriteBufferSize`

## HttpUserInfo

This tag specifies the physical location where the user-defined XML file is stored in the server.

### Description

By default the XML files are placed in the `uInfo` directory in the server installation directory.

When the `enable` attribute is set to "`true`", Flash Media Server responds to the HTTP request and returns the content of the XML file in the `uInfo` directory. The default setting for the `enable` attribute is "`false`".

Users can specify any XML file in the `uInfo` folder. The `uInfo` folder is configured to point to `C:\fms\uInfo`.

To get the XML file from the server, the HTTP request must begin with:

`http://server:port/fms/uInfo`

The syntax for an HTTP request for an XML file in `C:\fms\uInfo\foo\bar.xml` is:

`http://server:1935/fms/uInfo/foo/bar.xml`

## IdleAckInterval

This tag specifies the maximum time the server may wait before it sends back an `ack` (acknowledgement code) for a client idle post.

`Ack` is shorthand for acknowledgement code, a transmission control character used to indicate that a transmitted message was received uncorrupted or without errors. It also indicates that the receiving server is ready to accept transmissions. The receiver sends the code to the sender to indicate that the transmission has been accepted.

The values for this tag and the `IdlePostInterval` tag affect the latency observed by a client tunneling into the server. These tags should be configured at the same time.

### Description

The default settings for the `IdleAckInterval` and `IdlePostInterval` tags provide medium latency and is set to 512/512 milliseconds.

Low values reduce the latency but increase the network bandwidth overhead. Applications desiring low latency may configure the combination of values 128/256 for the `IdlePostInterval` and `IdleAckInterval` tags. Those applications not sensitive to high latencies may use the combination 1024/2048.

### See also

`IdlePostInterval`

## IdlePostInterval

This tag specifies in milliseconds the interval at which the client sends idle posts to the server to indicate that Flash Player has no data to send.

### Description

The default settings for the `IdleAckInterval` and `IdlePostInterval` tags provide medium latency and are set to 512/512 milliseconds.

Low values reduce the latency but increase the network bandwidth overhead. Applications desiring low latency may configure the combination of values 128/256 for `IdlePostInterval` and `IdleAckInterval` tags. Those applications not liable to high latencies can use the configuration 1024/2048.

### See also

`IdleAckInterval`

## MaxFailures

This tag specifies the maximum number of failures an edge server may incur before it restarts.

### Description

Default number of failures is 2.

## MaxSize

This tag specifies the maximum number of XML files cached in the server.

### Description

When the results from an HTTP request arrives from the origin server, the edge server loads the XML files into a cache. If another client requests the same file, the edge server sends it directly from the cache.

The default number of files in the cache is 100. Once the cache reaches the maximum size, the server checks for and removes the least-used 30% of the cache.

## MaxWriteDelay

The HTTP tunneling protocol ensures that a server will be able to write every four seconds. Occasionally, when connections close under abnormal conditions the notification may not reach the server, which may continue to place writes in a queue.

### Description

Anomalous connections are closed after the specified wait time. The default wait time is 40 seconds.

### Example

```
<Edge name="Edge1">
  <Enable>true</Enable>
  <IdlePostInterval>512</IdlePostInterval>
  <IdleAckInterval>512</IdleAckInterval>
  <MimeType>application/x-fms</MimeType>
  <WriteBufferSize>16</WriteBufferSize>
  <SetCookie>false</SetCookie>
  <RedirectHost secure="false">:8080</RedirectHost>
  <NeedClose>true</NeedClose>
  <MaxWriteDelay>40</MaxWriteDelay>
</Edge>
```

You may want to use this sample code as a template for configuring each edge server.

## MimeType

This tag specifies the default MIME (Multipurpose Internet Mail Extensions) type header sent on tunnel responses.

### Description

The server generally uses the MIME type specified by the incoming requests. The server will use the entry for the `MIMEType` tag only if it is unable to determine the MIME type from the incoming requests.

## NeedClose

This tag specifies whether or not HTTP 1.0 non-keepalive connections are to be closed once the response is written.

### Description

The default is to close the connections.

## NodeID

This tag specifies a unique node identification that supports the implementation of load balancers.

### Description

If the `NodeID` tag is used, a following string of up to 9 characters is prefixed to the tunnel session IDs and can be used by the load balancers to uniquely identify each node in the cluster.

The ID must contain URL safe characters except for '.' and '/', which are replaced by '_' and '-' respectively.

## Order

This tag specifies the sequence in which Flash Media Server evaluates the `Allow` and `Deny` tags.

### Description

```
<Order>Allow,Deny</Order>
```

The default sequence `Allow,Deny` indicates that access to a server is denied unless it is specified in the `Allow` tag

```
<Order>Deny,Allow</Order>
```

The alternative sequence `Deny,Allow` indicates that access to a server is allowed unless specified in the `Deny` tag and not specified in the `Allow` tag.

### See also

`Allow`, `Deny`

## Path

This tag specifies the location of the `uInfo` directory where the user-defined XML files are stored.

### Description

By default the `uInfo` directory is located in the server installation directory.

## RecoveryTime

This tag specifies the wait time for an edge server to pause after failing before restarting.

### Description

Once an edge server fails, it waits for the interval specified here before it restarts. The wait time is specified in seconds.

The number of failures is specified by the `MaxFailures` tag.

### See also

`MaxFailures`

## Redirect

This tag specifies whether or not the adaptor redirects unknown requests to an external server.

| NOTE | For redirection to work, HTTP tunneling must be enabled. |
|---|---|

### Description

An unknown request may connect only when it is the first request on a newly accepted connection. At any other time the request is considered an error and the connection is closed.

### Examples

```
<Host port="80">:8080</Host>
```

This example instructs Flash Media Server to redirect unknown requests to the specified redirect host.

```
<Host port="443">:8443</Host>
```

This example configures the `Redirect` tag to forward the request to a specific host depending upon which port the request arrived on.

```
<Redirect enable="false" maxbuf="16384">
```

The `maxbuf` attribute determines how big the IO buffers are. Flow control automatically handles the request when the bandwidth resources for producers and consumers differ widely. Flow control begins when the buffer in either direction fills up.

## ResourceLimits

Container tag.

### Description

The tags in this container configure the resource limits for the edge server.

### Contained tags

`MaxFailures`, `RecoveryTime`

## SetCookie

This tag specifies whether or not Flash Media Server sets a cookie.

### Description

Cookies are required when using load balancers to ensure that requests corresponding to one network connection are always sent to the same server. Keep in mind that the cookie adds to the HTTP header size and increases the bandwidth overhead.

## SSL

Container tag.

### Description

The tags in this section configure the incoming connections via the Secure Sockets Layer protocol, known as SSL. The `SSL` tags in Adaptor.xml configure Flash Media Server to act as an SSL-enabled server to accept incoming SSL connections.

You need to acquire a digital certificate to use SSL. Once you get your SSL certificate through a certificate authority such as Verisign, or by creating it yourself with a product such as OpenSSL, you then use the SSL tags to configure Flash Media Server for SSL.

The following is a quick-start to allowing SSL-enabled connections to Flash Media Server.

- Go to the `SSL` section of the Adaptor.xml file.
- Specify the location of the certificate in the `SSLCertificateFile` tag.
- Specify where to find the associated private key file in the `SSLCertificateKeyFile` tag.
- If the private key file is encrypted, specify the passphrase to use for decrypting the private key file in the `SSLPassPhrase` tag.
- Save the modified Adaptor.xml file.

### Contained tags

`SSLServerCtx` container.

## SSLCACertificateFile

This tag specifies the location of the certificate to return to clients who want to make a secure connection to the server.

### Description

If an absolute path is not specified, the certificate location is assumed to be relative to the adaptor directory.

### See also

`SSLCACertificateKeyFile`

## SSLCACertificateKeyFile

This specifies the location of the private key file that corresponds to the public key in the certificate specified in `SSLCertificateFile` tag.

### Description

If this file is encrypted, a password must be specified for decrypting, and placed in the `SSLPassPhrase` tag described below. If an absolute path to the key file is not specified, it is assumed to be relative to the adaptor directory.

### Example

```
<SSLCertificateKeyFile type="PEM"></SSLCertificateKeyFile>
```

The `type` attribute specifies the type of encoding used for the certificate key file. The encryption format is either PEM (Privacy Enhanced Mail) or ASN1 (Abstract Syntax Notation 1). The default is PEM.

### See also

SSLPassPhrase

## SSLCipherSuite

This tag specifies the suite of encryption ciphers that Flash Media Server uses to secure incoming connections.

### Description

This tag contains a list of colon-delimited components. A component can be a key exchange algorithm, authentication method, encryption method, digest type, or one of a selected number of aliases for common groupings.

```
<SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH</SSLCipherSuite>
```

NOTE

Contact Flash Media Server Technical Support before changing the default settings as listed in this example.

## SSLPassPhrase

This tag specifies the passphrase to use for encrypting the private key file.

### Description

This tag specifies the password to use for decrypting the key file if the key file is encrypted. If the key file is not encrypted, this tag is left blank.

To prevent plain text passwords appearing in the configuration file, this can be specified by doing a base64 encoding on the password and setting the `encrypt` attribute to `"true"`.

### Example

```
<SSLPassPhrase encrypt="true">dGluY2Fu</SSLPassPhrase>
```

The encrypted password is equivalent to the plaintext format:

```
<SSLPassPhrase>tincan</SSLPassPhrase>
```

or

```
<SSLPassPhrase encrypt="false" >tincan</SSLPassPhrase>
```

Even though the tag attribute is named `"encrypt"`, it is not a true encryption. It is a base64 encoding that makes the password less readable.

### See also

SSLCACertificateKeyFile

## SSLServerCtx

Container tag.

### Description

The tags in this section control the SSL configuration for this adaptor.

### Contained tags

`SSLCACertificateFile`, `SSLCACertificateKeyFile`, `SSLCipherSuite`, `SSLPassPhrase`, `SSLSessionTimeout`

## SSLSessionTimeout

This tag specifies in minutes how long an SSL-based session remains valid. The default time period is 5 minutes.

### Description

SSL sessions are used to improve performance by avoiding the need to perform the full SSL handshake for every connection. When a client connects to a server for the first time, it must perform the full handshake. After that first handshake, the server sends back a session object to the client which the client can place in the cache and reuse at a later time.

If the client connects to the same server again at a later time, it can send back the cached session object. The server will not require the full SSL handshake, if the session is still valid.

## UpdateInterval

This specifies how frequently the server checks the cache and updates its contents if the XML files have changed.

### Description

The default update interval is 5 seconds.

## WriteBufferSize

This tag specifies in kilobytes the size of the write buffer.

### Description

The default size is 16KBs.

# Vhost.xml file

The Vhost.xml configuration file defines an individual virtual host in Flash Media Server. Each virtual host directory on the server contains its own Vhost.xml file.

The Vhost.xml file contains tags that define the settings for the virtual host. These settings include aliases for the virtual host, the location of the virtual host's application directory, limits on the resources the virtual host can use, and other parameters.

Each virtual host must have its own directory inside the adaptor directory. The name of the directory must be the actual name of the virtual host, such as streaming.macromedia.com. Each defined virtual host must be mapped to a DNS [Domain Name Server] entry or another name resolution such as a WINS address or a hosts file, that specifies an IP address on the server computer.

Each adaptor must contain a _defaultVHost_ directory in addition to the custom virtual hosts that you define. If a client application tries to connect to a virtual host that does not exist, the server attempts to connect it to _defaultVHost_. If you are using a secure port for the adaptor that contains the virtual host, you can only define one virtual host to the adaptor, in addition to _defaultVHost_.

The Vhost.xml file contains the following tag structure.

```
<VirtualHost>
  <AliasList>
    <Alias name="acme">acme.myDomain.com</Alias>
  </AliasList>
  <AppsDir></AppsDir>
  <ResourceLimits>
    <MaxConnections></MaxConnections>
    <MaxAppInstances></MaxAppInstances>
    <MaxStreams>-1</MaxStreams>
    <MaxSharedObjects>-1</MaxSharedObjects>
    <AppInstanceGC>20</AppInstanceGC>
    <MessageCache>
      <MaxCacheUnits>4096</MaxCacheUnits>
      <MaxCacheSize>100</MaxCacheSize>
      <MaxUnitSize>16</MaxUnitSize>
      <FreeRatio>0.125</FreeRatio>
      <GlobalRatio>0.4</GlobalRatio>
      <MaxAge>1000000</MaxAge>
      <UpdateInterval>1024</UpdateInterval>
      <FreeMemRatio>0.5</FreeMemRatio>
    </MessageCache>
    <SmallMemPool>
      <MaxCacheUnits>4096</MaxCacheUnits>
      <MaxCacheSize>100</MaxCacheSize>
      <MaxUnitSize>16</MaxUnitSize>
      <FreeRatio>0.125</FreeRatio>
```

```
      <GlobalRatio>0.4</GlobalRatio>
      <MaxAge>1000000</MaxAge>
      <UpdateInterval>1024</UpdateInterval>
      <FreeMemRatio>0.5</FreeMemRatio>
    </SmallMemPool>
    <LargeMemPool>
      <MaxCacheUnits>4096</MaxCacheUnits>
      <MaxCacheSize>100</MaxCacheSize>
      <MaxUnitSize>16</MaxUnitSize>
      <FreeRatio>0.125</FreeRatio>
      <GlobalRatio>0.4</GlobalRatio>
      <MaxAge>1000000</MaxAge>
      <UpdateInterval>1024</UpdateInterval>
      <FreeMemRatio>0.5</FreeMemRatio>
    </LargeMemPool>
    <SegmentsPool>
      <MaxCacheUnits>4096</MaxCacheUnits>
      <MaxCacheSize>100</MaxCacheSize>
      <MaxUnitSize>16</MaxUnitSize>
      <FreeRatio>0.125</FreeRatio>
      <GlobalRatio>0.4</GlobalRatio>
      <MaxAge>1000000</MaxAge>
      <UpdateInterval>1024</UpdateInterval>
      <FreeMemRatio>0.5</FreeMemRatio>
    </SegmentsPool>
  </ResourceLimits>
  <VirtualKeys></VirtualKeys>
  <VirtualDirectory>
    <Streams></Streams>
  </VirtualDirectory>
  <DNSSuffix></DNSSuffix>
  <Allow></Allow>
  <Proxy>
    <Mode></Mode>
    <Anonymous></Anonymous>
    <CacheDir enabled="false"></CacheDir>
    <LocalAddress></LocalAddress>
    <RouteTable protocol="">
      <RouteEntry></RouteEntry>
    </RouteTable>
    <SSL>
      <SSLVerifyCertificate>true</SSLVerifyCertificate>
      <SSLCACertificatePath></SSLCACertificatePath>
      <SSLCACertificateFile></SSLCACertificateFile>
      <SSLVerifyDepth>9</SSLVerifyDepth>
      <SSLCipherSuite>ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH</SSLCipherSuite>
    </SSL>
  </Proxy>
</VirtualHost>
```

# Summary of Vhost.xml tags

The following list alphabetically displays the tags in the Vhost.xml configuration file.

| Vhost.xml tag | Description |
|---|---|
| Alias | Specifies the assumed name(s) for the virtual host. |
| AliasList | Container tag; contains the list of Alias tags. |
| Allow | Specifies the domains that can connect to this virtual host. |
| Anonymous | Determines whether or not this virtual host runs as an anonymous proxy. |
| AppInstanceGC | Specifies how often to check for and remove unused application instances. |
| AppsDir | Specifies the Applications directory for this virtual host. |
| CacheDir | Specifies the physical location where streams are cached on a proxy server. |
| DNSSuffix | Specifies the primary DNS (Domain Name Server) for this virtual host. |
| FreeMemRatio | Sets the maximum percentage of total memory that the total pool size may occupy. |
| FreeRatio | Specifies the percentage of the message cache to be consumed by the free list on a per-thread basis. |
| GlobalRatio | Specifies the percentage of the message cache to be consumed by the free list on a global basis. |
| LargeMemPool | Container tag; the tags in this section configure the small memory pool. |
| LocalAddress | Specifies a local IP Address for a proxy's outgoing connection. |
| MaxAge | Specifies the maximum reuse count before freeing the cache unit. |
| MaxAppInstances | Specifies the maximum number of application instances that can be loaded onto the virtual host. |
| MaxCacheSize | Specifies the maximum size of the cache. |
| MaxCacheUnits | Specifies the maximum number of free units in the cache. |
| MaxConnections | Specifies the maximum number of clients that can connect to this virtual host. |
| MaxSharedObjects | Specifies the maximum number of shared objects that can be created. |

| Vhost.xml tag | Description |
|---|---|
| MaxStreams | Specifies the maximum number of streams that can be created. |
| MaxUnitSize | Specifies the size threshold for messages that can be returned to the cache. |
| MessageCache | Container tag; tags in this section configure how messages are kept for reuse by Flash Media Server. |
| Mode | Configures this virtual host to run applications locally or remotely. |
| Proxy | Container tag; the tags in this section specify the settings for the virtual host to act as a proxy server and forward connection requests from applications to another Flash Media Server, and also behave locally as a remote server. |
| ResourceLimits | Container tag; the tags in this section specify the maximum resource limits for this virtual host. |
| RouteEntry | Maps the proxy's host:port pair to a different host:port pair. |
| RouteTable | Container tag; the tags in this section specifies the proxy's routing information. |
| SegmentsPool | Container tag; contains tags that configure how the segments pool caches segments of FLV (Flash Video) files. |
| SmallMemPool | Container tag; the tags in this section configure the large memory pool. |
| SSL | Container tag; the tags in this section configure this virtual host for secure communications. |
| SSLCACertificateFile | Specifies the name of a file that contains one or more CA certificates in PEM encryption format. |
| SSLCACertificatePath | Specifies the name of the directory containing one or more CA certificates. |
| SSLCipherSuite | Specifies the encryption ciphers for secure communications. |
| SSLClientCtx | Container tag; contains tags to configure Flash Media Server as an SSL (Secure Socket Layer) client for outgoing SSL connections. |
| SSLRandomSeed | Specifies the cryptographic accelerator to use. |
| SSLRandomSeed | Specifies the number of bytes of entropy to use for seeding the pseudo-random number generator (PRNG). |

| Vhost.xml tag | Description |
|---|---|
| SSLSessionCacheGC | Specifies how often to flush expired sessions from the server-side SSL session cache. |
| SSLVerifyCertificate | Specifies whether or not to verify the certificate returned by the server being connected to. |
| SSLVerifyDepth | Specifies the maximum depth in the certificate chain that Flash Media Server is willing to accept. |
| Streams | Specifies the virtual directory for recorded streams. |
| UpdateInterval | Specifies how often thread statistics are collected per reused messages. |
| VirtualDirectory | Container tag; configures the directory mappings for resources such as recorded streams. |
| VirtualHost | Root tag; contains all other tags for the Vhost.xml file. |
| VirtualKeys | Sets the virtual key mappings for connecting players. |

# Description of Vhost.xml tags

The following alphabetical list of Vhost.xml tags contains additional information, including cross references to associated tags, syntax, and examples.

## Alias

The `Alias` tag specifies the assumed name(s) of the virtual host.

### Description

`An alias is` an alternative short name to use when connecting to the virtual host. The `Alias` tag lets you specify additional names to connect to this virtual host. Use the `Alias` tag to shorten long host names, or if you want to be able to connect to this virtual host with different names.

### Syntax

```
<Alias name="alias1"></Alias>
```

### Example

```
<Alias name="abc">abc.macromedia.com</Alias>
```

If the name of this virtual host is "abc.macromedia.com", but you wish to connect by simply specifying "abc", then specify the alias "abc". Keep in mind that "abc" must still map to the same IP address as "abc.macromedia.com".

If more than one virtual host on the same adaptor has been defined with the same alias, then the first match that is found is taken. You can avoid unexpected behavior by specifying a unique alias for each virtual host.

### See also

`AliasList`

## AliasList

Container tag

### Description

The tags nested in this section list the alias(es) for this virtual host. You can specify an unlimited number of aliases by adding additional `Alias` tags. Each `Alias` must map to the IP address for the virtual host.

### Contained tag

`Alias`

## Allow

This tag is a comma-delimited list of domains that are allowed to connect to this virtual host.

### Description

If the `Allow` tag is left empty, the only connections allowed are those coming from the same domain.

### Examples

```
<Allow>macromedia.com,yourcompany.com</Allow>
```

This example allows only connections from the macromedia.com and yourcompany.com domains.

```
<Allow>localhost</Allow>
```

This example allows `localhost` connections only.

```
<Allow>all</Allow>
```

This example allows connections from all domains. Macromedia does not recommend the use of "`all`"; it may create a security risk.

## Anonymous

This tag configures the virtual host as an anonymous proxy (also called an implicit or transparent proxy) or as an explicit proxy.

### Description

Both anonymous and explicit proxies intercept and aggregate the clients' requests to connect to the origin server. Here are some key differences between anonymous and explicit proxies:

- The identity (IP address and port number) of an anonymous server is hidden from the client.
- The anonymous proxy does not change or modify the routing information in the incoming URI before connecting the client(s) to the origin server.
- The URI for an explicit proxy specifies the edge server(s) that will intercept connection requests to the origin server.

You can create a chain of proxies by specifying them in the URI.

- Any anonymous proxy in the chain passes on without modification the routing information in the URI to the next edge server in the chain.
- The routing information in the URI for a chain of explicit proxies specifies the edge servers that are chained together to intercept connection requests to the origin server.
- The routing information in the URI for a chain of explicit proxies specifically identifies the sequence of edge servers in the chain.
- The URI for a chain of explicit proxies directs all clients' connection requests through a specific sequence of edge servers before making the connection to the origin server.
- The explicit proxy modifies the routing information in the URI by stripping off its token or identifier in the URI before passing the URI on to the next server in the chain.

### Syntax

```
<Anonymous>false</Anonymous>
```

The default value is `false`. Setting this tag to `true` creates an implicit proxy to intercept the incoming URIs.

### See also

`Mode`

## AppInstanceGC

This tag specifies how often to check for and remove unused resources for application instances, such as Shared Objects, Streams, and Script engines.

### Description

The default interval is 1 minute.

## AppsDir

This tag specifies the Applications directory for this virtual host.

### Description

The Applications directory is the base directory where all applications for this virtual host are defined. You define an application by creating a directory with the application name.

- In Windows, the default `AppsDir` location is `C:\Program Files\Macromedia\Flash Media Server\applications`.
- On Linux, the default location is `/opt/macromedia/fms/applications`.

### Example

```
<AppsDir>C:\MyApps;D:\NewApps</AppsDir>
```

You can also specify multiple applications directories by separating locations with a semicolon (;). You can specify two locations, each of which contains application subdirectories. If you change the default location of the `AppsDir` tag, be sure to include a directory named `admin` in each directory. This ensures that the management console (`fmsconsole.swf`) will be able to connect to the virtual host.

If no location is specified for this tag, the applications directory is assumed to be located in the `vhost` directory.

For more information, see "Using the management console" on page 16.

## CacheDir

This tag enables or disables writing recorded streams to disk. Set this tag on the proxy or edge server to control the proxy's caching behavior.

### Description

The contents of the cache are volatile. This tag controls whether the cached streams will be written to disk, in addition to being cached in the proxy server's memory.

The proxy server caches content locally to aid performance. Caching static content can reduce the overall load placed on the origin server.

The default location is the cache folder in the server installation directory.

### Syntax

```
<CacheDir enabled="false"></CacheDir>
```

The default value of the `enable` attribute is "`false`".

### Example

```
<CacheDir enabled="true">c:\mycache</CacheDir>
```

To save the contents of the cache, set the `enable` attribute to "`true`" and specify a directory on the disk where the files will be written.

### See also

`CachePrefix` in the Application.xml file

## DNSSuffix

This tag specifies the primary DNS suffix for this virtual host.

### Description

If a reverse DNS lookup fails to return the domain as part of the host name, then this tag is used as the domain suffix.

## FreeMemRatio

Located in `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

This tag specifies the maximum percentage of total memory that the total pool size may occupy.

### Description

This tag's setting ranges between 0 and 1. The default setting is 0.5.

### See also

`FreeRatio`, `GlobalRatio`

## FreeRatio

Located in LargeMemPool, MessageCache, SegmentsPool, and SmallMemPool containers.

This tag specifies the percentage of the message cache to be consumed by the free list on a per-thread basis.

### Description

This tag's setting ranges between 0 and 1. The default setting is 0.125. When more free memory is available to a thread than the specified ratio, the freed memory will return to the global pool.

### See also

FreeMemRatio, GlobalRatio

## GlobalRatio

Located in LargeMemPool, MessageCache, SegmentsPool, and SmallMemPool containers.

This tag specifies the percentage of the message cache to be consumed by the free list on a global basis.

### Description

This tag's setting ranges between 0 and 1. Its default setting is 0.4. When more free memory is available to a thread than the specified ratio, the freed memory will return to the operating system.

### See also

FreeMemRatio, FreeRatio

## LargeMemPool

Container tag.

### Description

The Large Memory Pool caches large chunks of memory within Flash Media Server to increase performance of large allocations.

### Contained tags

FreeMemRatio, FreeRatio, GlobalRatio, MaxAge, MaxCacheSize, MaxCacheUnits, MaxUnitSize, UpdateInterval

## LocalAddress

This tag binds a proxy's outgoing connection to a specific local IP address.

### Description

The `LocalAddress` tag lets you allocate incoming and outgoing connections to different network interfaces. This strategy is useful when configuring a proxy to either transparently pass on or intercept requests and responses.

If the `LocalAddress` tag is not specified, then outgoing connections bind to the value of the `INADDR_ANY` Windows system variable.

## MaxAge

Located in `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

This tag specifies the maximum reuse count before freeing the cache unit.

### Description

The default reuse count is 1000000.

## MaxAppInstances

This tag specifies the maximum number of application instances that can be loaded into this virtual host.

### Description

A chat application, for example, might require more than one instance, because each chat room represents a separate instance of the application on the server. The default number is 15000 application instances.

A Flash SWF file defines which application instance it is connecting to by the parameters it includes with its ActionScript `connect` call.

## MaxCacheSize

Located in `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

This tag specifies the size of the cache in megabytes.

### Description

The default cache size is 100 megabytes.

### See also

`MaxCacheUnits`

## MaxCacheUnits

Located in `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

This tag specifies the maximum number of free units in the cache.

### Description

The default number of free units is 4096. Note the number of free units may be less if the size limit specified by the `MaxCacheSize` tag is reached.

### See also

`MaxCacheSize`

## MaxConnections

This tag specifies the maximum number of clients that can connect to this virtual host.

### Description

The maximum number of allowed connections is encoded in the license file. Connections are denied if the specified limit is exceeded.

The default number is -1, which represents an unlimited number of connections.

## MaxSharedObjects

This tag specifies the maximum number of shared objects that can be created.

### Description

The default number of shared objects is 50000.

## MaxStreams

This tag specifies the maximum number of streams that can be created.

### Description

The default number of streams is 250000.

## MaxUnitSize

Located in `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

This tag specifies the size threshold for messages that can be returned to the cache.

### Description

The threshold is specified in kilobytes. The default threshold size is 16 kilobytes.

## MessageCache

Container tag.

### Description

This section contains the tags that control how the message cache holds onto messages used in the Flash Media Server system. The message cache retains the messages in memory for reuse instead of returning them and repeatedly requesting them from the operating system.

Messages are the essential communication units of Flash Media Server and recycling them improves Flash Media Server performance.

### Contained tags

`FreeMemRatio`, `FreeRatio`, `GlobalRatio`, `MaxAge`, `MaxCacheSize`, `MaxCacheUnits`, `MaxUnitSize`, `UpdateInterval`

## Mode

The `Mode` tag configures whether Flash Media Server runs locally as an origin server or remotely as an edge server.

### Description

The `Mode` tag can be set to `local` or `remote`. The default setting is `local`.

- When the Mode tag is set to `local`, the Flash Media Server runs its applications locally and is called an origin server.
- When the `Mode` tag is set to `remote`, the server behaves as a proxy or edge server that connects to the applications running on an origin server.
- If the `Mode` tag is undefined, the virtual host is evaluated as an alias for the default virtual host and assumes its configuration.

### Syntax

```
<Mode>local</Mode>
```

### See also

`Anonymous`, `Proxy`

## Proxy

Container tag.

### Description

The tags nested in this section configure this virtual host as a proxy server that can forward connection requests from applications running on one remote server to another remote server.

| NOTE | Whenever a virtual host is configured as a proxy server, it behaves locally as a remote server. |
|---|---|

If this virtual host is configured to run in `remote` mode and you want to configure the properties of an outgoing SSL connection to an upstream server, the SSL connection to upstream servers will use the default configuration specified in the `SSL` section of the Server.xml file.

For more information on this section of the Server.xmlfile, see "SSL" on page 119.

### Contained tags

`Anonymous`, `CacheDir`, `LocalAddress`, `Mode`, `RouteTable`, `SSL`

## ResourceLimits

Container tag.

### Description

The tags in this section specify the maximum resources limits for this virtual host.

### Contained tags

`LargeMemPool`, `MessageCache`, `SegmentsPool`, `SmallMemPool` containers

`AppInstanceGC`, `MaxAppInstances`, `MaxConnections`, `MaxSharedObjects`, `MaxStreams` tags

## RouteEntry

This tag contains the routing information that instructs the proxy to forward the connection request to one server's IP address and port number [host:port] to a different `host:port`.

### Syntax

```
<host1>:<port1>;<host2>:<port2>
```

The syntax instructs a Flash Media Server proxy where to route the connection to `host1:port1` to `host2:port2`.

### Description

Proxies or edge servers are configured with the `RouteEntry` tag to direct connections to another destination. The `RouteTable` tag contains the `RouteEntry` tags that control where the edge or proxy server reroutes requests.

You can also add the `protocol` attribute to an individual `RouteEntry` tag to specify how the the edge or proxy server reroutes requests. If no protocol is specified, however, Flash Media Server applies the protocol specified in the `RouteTable` tag. Implicit proxies hide the routing information from the clients.

The connection syntax for this tag is flexible, as demonstrated in the following examples.

### Examples

```
<Proxy>
  <RouteTable protocol="">
  <RouteEntry>foo:1935;bar:80</RouteEntry>
  </RouteTable>
</Proxy>
```

This example shows how you can configure the proxy to route all connections to the host "`foo`" to the host "`bar.`"

```
<RouteEntry>*:*;foo:1935</RouteEntry>
```

Flash Media Server allows the use of the wildcard character '*' to replace host and port.The example shows how to route connections destined for any host on any port to port 1935 on the host "`foo.`"

```
<RouteEntry>*:*;*:1936</RouteEntry>
```

The example instructs Flash Media Server to route connections to any host on any port, to the specified host on port 1936. For example, if you were to connect to `"foo:1935"`, the connection would be routed to `"foo:1936"`.

```
<RouteEntry>*:*;*:80</RouteEntry>
```

The example instructs Flash Media Server to use the values for host and port on the left side as the values for host and port on the right side, and to route connections destined for any host on any port to the same host on port 80.

```
<RouteEntry>foo:80;null</RouteEntry>
```

The example instructs Flash Media Server to route a host:port combination to `null`. Its effect is to reject all connections destined for `foo:80`.

### See also

`Proxy` container, `Anonymous`, `Mode`, `RouteTable` tags

# RouteTable

Container tag.

## Description

The `RouteEntry` tags nested under the `RouteTable` tag specify the routing information for the proxy or edge server. Administrators use the these tags to route connections to the desired destination. The `RouteTable` tag can be left empty or it can contain one or more `RouteEntry` tags.

The `protocol` attribute specifies the protocol to use for the outgoing connection. The attribute is set to `""`, `"rtmp"` for a non-secure connection, or `"rtmps"` for a secure connection.

## Syntax

```
<RouteTable protocol="rtmp">
```

or

```
<RouteTable protocol="rtmps">
```

- Specifying `""` means preserving the security status of the incoming connection.
  - If the incoming connection was secure, than the outgoing connection will also be secure.
  - If the incoming connection was non-secure, the outgoing connection will be non-secure.
- Specifying `"rtmp"` instructs the proxy or edge to use a non-secure outgoing connection, even if the incoming connection was secure.
- Specifying "rtmps" instructs the proxy or edge to use a secure outgoing connection, even if the incoming connection was non-secure.

You can override the security status for a connection mapping by specifying a `protocol` attribute in a `RouteEntry` tag. By default, Flash Media Server applies the protocol configured in the `RouteTable` list unless the mapping for a particular `RouteEntry` tag overrides it.

## Contained tag

`RouteEntry`

## SegmentsPool

Container tag.

### Description

The tags nested within this container configure how the segments pool caches segments of FLV (Flash Video) files within Flash Media Server to increase performance of FLV streaming and keep frequently used FLV files available in memory.

### Contained tags

`FreeMemRatio`, `FreeRatio`, `GlobalRatio`, `MaxAge`, `MaxCacheSize`, `MaxCacheUnits`, `MaxUnitSize`, `UpdateInterval`

## SmallMemPool

Container tag.

### Description

The tags nested within this container configure the small memory pool. The small memory pool saves small chunks of memory within Flash Media Server to increase performance of small allocations

### Contained tags

`FreeMemRatio`, `FreeRatio`, `GlobalRatio`, `MaxAge`, `MaxCacheSize`, `MaxCacheUnits`, `MaxUnitSize`, `UpdateInterval`

### See also

`LargeMemPool`, `MessageCache`, `SegmentsPool`

## SSL

Container tag.

### Description

If a virtual host is running in `remote` mode as a proxy or edge server and you want to configure the properties of an outgoing SSL connection to an upstream server, then you must enable this section and configure its SSL tags appropriately.

When Flash Media Server acts as a client to make an outgoing SSL connection, the following sequence of events takes place:

- The `SSL` tags in the Vhost.xml file are evaluated first.

- If the SSL tags in the Vhost.xml file override the SSL tags in the Server.xml file, Flash Media Server uses the SSL tags in the Vhost.xml file to configure the connection.
- If the SSL tags in the Vhost.xml file match the SSL tags in the Server.xml file, Flash Media Server uses the default values for SSL in the Server.xml file to configure the connection.
- If the SSL tags in a proxy's Vhost.xml file are not present, Flash Media Server uses the default values specified in the SSL section of Server.xml to configure the SSL connection to upstream servers.

| NOTE | When Flash Media Server is running in local mode as an origin server, the SSL information in the vhost.xml file is not evaluated. |
|---|---|

You can also override the configuration for outgoing SSL connections for an individual virtual host in Vhost.xml by copying the SSL tags in Server.xml to the corresponding SSL section in the Vhost.xml file.

For more information on the SSL tags in Server.xml, see "SSL" on page 119.

### Contained tags

SSLClientCtx container

SSLRandomSeed, SSLRandomSeed, SSLSessionCacheGC tags

### See also

SSLClientCtx

## Streams

This tag specifies the virtual directory mapping for recorded streams.

### Description

You can specify multiple virtual directory mappings for streams by adding additional Streams tags—one for each virtual directory mapping.

### Example

```
<Streams>foo;c:\data</Streams>
```

This example maps all streams whose names begin with foo/ to the physical directory c:\data. The stream named foo/bar would map to the physical file c:\data\bar.flv.

If there is a stream named foo/bar/x then Flash Media Server first tries to find a virtual directory mapping for foo/bar. If there is no virtual directory for foo/bar, Flash Media Server then checks for a virtual directory mapping for foo. Since a virtual directory mapping does exist for foo, the stream foo.bar maps to the file c:\data\bar\x.flv.

### Syntax

*virtualDirectory*;*actualDirectory*

The Streams tag allows you to specify a virtual directory for stored stream resources used by more than one application. By using a virtual directory, you specify a relative path that points to a shared directory that multiple applications can access.

### Example

```
<Streams>common;C:\FlashMediaServer\myApplications\shared/resources\
  </Streams>
```

If the virtual directory you specify does not end with a backward slash, one is added by the server.

Any application that refers to a stream whose path begins with common/ will access the item in C:\FlashMediaServer\myApplications\shared\resources regardless of the application's own directory structure. If the application VideoConference refers to an item common/video/recorded/June5 and the application Collaboration refers to common/videorecorded/June5, they both point to the same item C:\FlashMediaServer\myApplications\shared\resources\video\recorded\June5\.

### Additional information

This release includes a feature called custom stream delivery, which allows Flash Media Server to match the decoding of a recorded stream to the version of Flash Player on the client. Custom stream delivery allows an application running on Flash Media Server to switch between the Sorenson and On2 CODECs (data COmpressor/DECompressor) without changes to the code.

Custom stream delivery is enabled using a combination of the VirtualDirectory and VirtualKey tags, and the Client.virtualKey and Stream.setVirtualPath() APIs.

For optimal results:

- Streams replaying on Flash Players 6 and 7 use the Sorenson technology.
- Streams replaying on Flash Player 8 use the On2 technology.

For instance, the application developer might locate a stream encoded with the On2 codec in one folder and create a different folder for the same stream encoded with the Sorenson codec. Both streams have the same content, but each is tailored to replay on specific versions of Flash Player.

You specify more than one virtual directory mapping by adding multiple Streams tags.

### Example

```
<VirtualDirectory>
  <Streams key="A">foo;c:\streams\on2</Streams>
  <Streams key="B">foo;c:\streams\sorenson</Streams>
  <Streams key="">foo;c:\streams</Streams>
</VirtualDirectory>
```

The `Key` attribute matches the key found in the Flash Player string `FlashVer` on the client, which is automatically sent to Flash Media Server with the login.

The administrator sets the key. The `client` object in the server-side script has the writable `client.virtualKey` property. Flash Player 8 and Flash Player 9 are assigned `Key A`; earlier versions of the player get `Key B`. When the client plays a stream, it will use the appropriate key. The replay of a recorded stream fails if the player does not find a key.

### See also

`VirtualDirectory`

## UpdateInterval

Located in `LargeMemPool`, `MessageCache`, `SegmentsPool`, and `SmallMemPool` containers.

This tag specifies how frequently thread statistics are collected for reused messages.

### Description

The default count is 1024 messages.

## VirtualDirectory

This tag contains the virtual directory mappings for resources such as recorded streams.

### Description

Virtual directories let you share resources such as recorded streams across different applications. Virtual directories let multiple applications share the resources located in an application directory. When the beginning portion of a resource's URI matches a specific virtual directory, Flash Media Server maps the storage location of the resource to the virtual directory.

If you are mapping a virtual directory to a drive on another computer, make sure that the computer running Flash Media Server has the right permissions to access the other computer.

### Syntax

```
<VirtualDirectory>
  <Streams>key-value;virtual path;directory</Streams>
</VirtualDirectory>
```

You specify a virtual directory by mapping the client's virtual key to the resource's actual key. Setting the key to point to the beginning of the value for the `Streams` tag forces the virtual directory mapping in a `Streams` to use the original mapping in the original `Streams` tag.

### Example

```
<VirtualDirectory>
  <Streams>alphaKey;foo;c:\goodStreams</Streams>
  <Streams>betaKey;foo;c:\evenBetterStreams</Streams>
</VirtualDirectory>
```

You create a stream switching scenario by including more than one copy of the same virtual path with different keys. This example shows two pointers to the same virtual path using different keys.

### Additional information

Custom stream delivery is a new feature in Flash Media Server 2. This task that is enabled using a combination of the `VirtualDirectory` and `VirtualKey` tags, and the `Client.virtualKey` and `Stream.setVirtualPath()` APIs.

### Contained tag

`Streams`

### See also

`VirtualKeys`

## VirtualHost

Root tag of the Vhost.xml file.

### Description

This tag contains all the configuration tags for Vhost.xml

## VirtualKeys

This tag sets the virtual key mappings for the different versions of Flash Player connecting to Flash Media Server. This tag and the `VirtualDirectory` tag implement the custom stream delivery feature in Flash Media Server 2.

### Description

When the Flash Player running on a client connects to Flash Media Server, it receives a virtual key. This tag sets up which Flash Player versions are mapped to a particular key. A range of Flash Player versions is matched according to the `FlashVer` string that the player automatically sends to the server.

```
<VirtualKeys>
  <Key from="WIN 8,0,0,0" to="WIN 9,0,0,0">A</Key>
  <Key from="WIN 6,0,0,0" to="WIN 7,9,9,9">B</Key>
  <Key from="MAC 6,0,0,0" to="MAC 7,0,55,0">B</Key>
</VirtualKeys>
```

This example shows how the key can be set by the administrator as a client property in the server-side script. If the client has key A, the `Key` attribute will map to on2, and if it has key B it will map to Sorenson.

```
<VirtualKeys from="WIN 7,0,19,0" to="WIN 9,0,0,0">A</VirtualKeys>
```

This example shows if the `VirtualKey` tag is not specified for a version of Flash Player, Flash Media Server applies a default setting.

### Additional information

Custom stream delivery is enabled using a combination of the `VirtualDirectory` and `VirtualKey` tags, and the `Client.virtualKey` and `Stream.setVirtualPath()` APIs.

### See also

`VirtualDirectory`

# Application.xml file

The Application.xml file contains the settings for the applications that will run on the server. These settings include, for example, the size of the Server-Side Media ActionScript runtime engine, the location at which streams and shared objects are stored, and bandwidth limitations.

Each virtual host can contain multiple Application.xml files. The Application.xml file in the virtual host directory configures the default settings for applications within the virtual host. If you want to have different settings for a particular application, create a specific Application.xml file in the application's registered application directory (for example, ../applications/*app_name*) with the settings you want.

## Overriding default settings

Flash Media Server administrators can include the optional `override` attribute for certain tags in the virtual host's Application.xml file.

### Example

```
<LoadOnStartup override="no">false</LoadOnStartup>
```

By default, the `Bandwidth` and `BandwidthCap` container tags includes an `override` parameter set to "`yes`", which allows the values for the `ClientToServer` and `ServerToClient` tags nested in these sections to be overridden. The `Client` tag in this XML files includes an `override="no"` attribute by default.

Here are the rules Flash Media Server uses when applying the `override` attribute:

- When the `override` attribute is included in a tag and set to `no`, application-specific Application.xml files can not override that tag's setting.
- All subtags under the `LoadOnStartup` tag cannot be overridden.
- If you omit the `override` attribute, the `LoadOnStartup` tag can be overridden.

The Application.xml file contains the following file structure.

```
<Application>
  <Process>
    <Scope></Scope>
    <LifeTime>
      <RollOver></RollOver>
      <MaxCores></MaxCores>
    </LifeTime>
    <MaxFailures></MaxFailures>
    <RecoveryTime></RecoveryTime>
  </Process>
  <LoadOnStartup>false</LoadOnStartup>
  <MaxAppIdleTime>1200</MaxAppIdleTime>
  <JSEngine>
    <RuntimeSize>1024</RuntimeSize>
    <MaxTimeOut>0</MaxTimeOut>
    <ScriptLibPath></ScriptLibPath>
    <FileObject>
      <VirtualDirectory></VirtualDirectory>
    </FileObject>
  </JSEngine>
```

```
<StreamManager>
  <StorageDir></StorageDir>
  <DuplicateDir></DuplicateDir>
  <CachePrefix></CachePrefix>
  <CacheUpdateInterval></CacheUpdateInterval>
  <EnhancedSeek>false</EnhancedSeek>
  <KeyFrameInterval>1000</KeyFrameInterval>
  <Audio>
    <CombineSamples>
      <Subscribers></Subscribers>
      <LoCPU></LoCPU>
      <HiCPU></HiCPU>
      <MaxSamples></MaxSamples>
    </CombineSamples>
    <SendSilence>
      <Interval></Interval>
    </SendSilence>
    <NotifyAudioStop>
      <Duration></Duration>
    </NotifyAudioStop>
  </Audio>
</StreamManager>
<SharedObjManager>
  <StorageDir></StorageDir>
  <DuplicateDir></DuplicateDir>
  <ResyncDepth></ResyncDepth>
  <LockTimeOut></LockTimeOut>
  <AutoCommit></AutoCommit>
</SharedObjManager>
<AllowHTTPTunnel></AllowHTTPTunnel>
<Client>
  <Bandwidth>
    <ServerToClient>250000</ServerToClient>
    <ClientToServer>250000</ClientToServer>
  </Bandwidth>
  <BandwidthCap override="no">
    <ServerToClient>10000000</ServerToClient>
    <ClientToServer>10000000</ClientToServer>
  </BandwidthCap>
  <HTTPTunnel>
    <IdlePostInterval>512</IdlePostInterval>
    <IdleAckInterval>512</IdleAckInterval>
    <MimeType></MimeType>
    <WriteBufferSize></WriteBufferSize>
  </HTTPTunnel>
  <MaxMessageSizeLosslessVideo>0</MaxMessageSizeLosslessVideo>
  <Access>
    <FolderAccess></FolderAccess>
  </Access>
  <UserAgent>
```

```
      <Bits></Bits>
    </UserAgent>
  </Client>
  <HTTP>
    <HTTP1_0></HTTP1_0>
    <Verbose></Verbose>
    <Connections>
      <MaxTimeout></MaxTimeout>
      <Reuse></Reuse>
      <Interface></Interface>
    </Connections>
    <Proxy>
      <Host></Host>
      <Port></Port>
      <Type></Type>
      <Tunnel></Tunnel>
      <Username></Username>
      <Password></Password>
    </Proxy>
    <Redirect>
      <Allow></Allow>
      <Max></Max>
      <UnrestrictedAuth></UnrestrictedAuth>
    </Redirect>
  </HTTP>
</Application>
```

# Summary of Application.xml tags

The following list alphabetically displays the tags in the Application.xml configuration file.

| Application.xml tag | Description |
|---|---|
| `Access` | Container tag; contains tag that controls the permission levels in the Access Module (the libconnect.dll file). |
| `Allow` | Allows or disallows the "follow and location:" header added with HTTP redirection. |
| `AllowHTTPTunnel` | Configures Flash Media Server to allow tunneling connections into this application. |
| `Application` | Root tag; this tag contains all tags in Application.xml. |
| `Audio` | Container tag; contains tags to configure the audio stream settings. |
| `AutoCommit` | Enables or disables the Shared Object Manager to automatically commit shared objects. |

| Application.xml tag | Description |
|---|---|
| Bandwidth | Container tag; contains tags to configure the bandwidth settings for server-client communications. |
| BandwidthCap | Container tag; contains tags that specify the maximum bandwidth values that a user can set. |
| Bits | Contains the settings for different versions of Flash Player on the Windows and Macintosh platforms. |
| CachePrefix | Specifies the cache prefix that is passed from the origin server to the proxy server. |
| CacheUpdateInterval | Specifies the interval for updating cache streaming in the proxy server. |
| Client | Container tag; contains tags to configure the client. |
| ClientToServer (Bandwidth) Bandwidth container | Specifies the bandwidth settings for client-to-server communications. |
| ClientToServer (BandwidthCap) BandwidthCap container | Specifies the bandwidth settings for client-to-server communications that can be set by the user. |
| CombineSamples | Container tag; contains tags to configure how Flash Media Server uses sound sampling. |
| Connections | Container tag; contains tags to configure settings for HTTP connections. |
| DuplicateDir (SharedObjManager) SharedObjManager container | Specifies the backup location for shared objects |
| DuplicateDir (StreamsManager) StreamManager container | Specifies the backup location for stream files. |
| Duration | Specifies the wait time before Flash Media Server notifies clients when audio stops in a stream. |
| EnhancedSeek | Enables the fine tuning of the seeking performance within streams by creating a key frame. |
| FileObject | Container tag; contains tag with file object setting. |
| FolderAccess | Configures folder-level permissions for the readAccess and writeAccess functions in the Access Module. |
| HiCPU | Specifies the upper limit to begin sound sampling. |
| Host | Specifies the HTTP proxy to use. |
| HTTP | Container tag; contains tags to configure the HTTP connections for this application. |

| Application.xml tag | Description |
|---|---|
| HTTP1_0 | Allows or disallows use of the HTTP 1.0 protocol. |
| HTTPTunnel | Container tag; contains tags to configure HTTP tunneling. |
| IdleAckInterval | Specifies the wait time before Flash Media Server responds to an idle post sent to it. |
| IdlePostInterval | Specifies the wait time before Flash Player sends an idle post message to Flash Media Server. |
| Interface | Specifies the name to use as the outgoing network interface. |
| Interval | Specifies the interval for sending silence messages when no audio is being published to a live stream. |
| JSEngine | Container tag; the tags in this section configure the JavaScript engine. |
| KeyFrameInterval | Specifies the time interval for saving keyframe in a FLV file. |
| LifeTime | Specifies the lifetime of stateless core processes. |
| LoadOnStartup | Specifies whether or not to load this application when the server starts. |
| LockTimeout | Specifies the time-out value before unlocking a shared object file. |
| LoCPU | Specifies the lower limit to halt sound sampling. |
| Max | Specifies the maximum number of HTTP redirections allowed. |
| MaxAppIdleTime | Specifies the maximum time an application instance can be idle. |
| MaxCores | Specifies the maximum number of core processes for an application. |
| MaxFailures | Specifies the maximum number of failures for a core process. |
| MaxMessagesizeLosslessvideo | Specifies the maximum size of messages for screen-sharing packets. |
| MaxSamples | Specifies the maximum number of samples that can be combined into a message. |
| MaxTimeOut (Connections)<br>Connections container | Specifies the maximum time for a transfer to be completed. |
| MaxTimeOut (JSEngine)<br>JSEngine container | Specifies the maximum time a script can take to execute a Java server function. |

| Application.xml tag | Description |
|---|---|
| MimeType | Specifies the default MIME-type header sent on tunnel responses. |
| NotifyAudioStop | Specifies whether Flash Media Server is notified when an audio transmission ending on a stream is encountered. |
| Password | Specifies the password for connections to the proxy. |
| Port | Specifies the proxy port to connect to if not specified. |
| Process | Container tag; contains tags to configure the process and recovery settings for applications. |
| Proxy | Container tag; contains tags to configure the HTTP proxy. |
| RecoveryTime | Specifies the recovery time for a core. |
| Redirect | Container tag; contains tags to configure HTTP redirection. |
| ResyncDepth | Specifies the resyncing interval for shared object files. |
| Reuse | Specifies whether or not to close the HTTP connection after each transfer. |
| RollOver | Specifies the time length a core process is in use. |
| RuntimeSize | Specifies the maximum size for the script engine. |
| Scope | Specifies the process scope in which the application runs. |
| ScriptLibPath | Contains a list of paths the Java Server engine can search to resolve a script file. |
| SendSilence | Container tag; contains tags to configure the sending of silence messages. |
| ServerToClient (Bandwidth)<br>Bandwidth container | Specifies the bandwidth settings for server-to-client communications. |
| ServerToClient (BandwidthCap)<br>BandwidthCap container | Specifies the maximum bandwidth a user can set for data sent from the server to the client. |
| SharedObjManager | Container tag; contains tags to configure the Shared Object Manager of an application. |
| StorageDir (StreamManager) | Specifies the locations where recorded streams and shared objects are stored. |
| StreamManager | Container tag; contains the Stream Manager settings for the Application. |
| Subscribers | Specifies a base number of subscribers required before implementing sound sampling. |

| Application.xml tag | Description |
|---|---|
| Tunnel | Specifies whether or not to tunnel all operations through a given HTTP proxy. |
| Type | Specifies the type of proxy being connected to. |
| UnrestrictedAuth | Allows or disallows sending username/password with each HTTP redirection. |
| UserAgent | Specifies the version dependency settings for clients that use different versions of Flash Player or platform. |
| Username | Specifies the username for connections to the proxy. |
| Verbose | Enables or disables the use of verbose information during HTTP operations. |
| WriteBuffSize | Specifies the size of the write buffer. |

# Description of Application.xml tags

The following alphabetical list of Application.xml tags contains additional information, including cross references to associated tags, syntax, and examples.

## Access

Container tag.

### Description

The Access Module consists of the libconnect.dll file. It intercepts and examines each connection request to Flash Media Server to determine whether the connection should be accepted or rejected.

The `FolderAccess` tag in this section sets permission levels in the Access Module.

### Contained tag

`FolderAccess`

## Allow

This tag specifies whether or not to allow the "`following and Location`:" header that is sent with redirection of an HTTP header.

### Description

The default is `true`, allowing HTTP redirects.

## AllowHTTPTunnel

The tag configures Flash Media Server to allow HTTP tunneling connections into this application.

### Description

By default, Flash Player communicates with Flash Media Server using the RTMP protocol over port 1935. If that fails, it will try again over ports 443 and 80 in an attempt to get around firewall settings, which prevents TCP/IP connections over non-standard ports.

In some cases, the Flash Player has to negotiate a connection to Flash Media Server through a proxy server, or use the HTTP protocol to transmit RTMP packets (called HTTP tunneling) if there is a firewall that allows only HTTP content to be sent out to public servers.

The values for this tag are described in the following table:

| Value | Description |
|---|---|
| `true` | Allows tunneling connections. |
| `false` | Disallows tunneling connections. |
| `http1.1only` | Allows HTTP 1.1 connections only. |
| `keepalive` | Allows HTTP 1.0 and 1.1 keepalive connections. |

## Application

Root tag.

### Description

This is the root tag for Application.xml. It contains all the other tags.

## Audio

Container tag.

### Description

The tags in this section specify the settings for audio streams on Flash Media Server.

### Contained tags

`CombineSamples`, `NotifyAudioStop`, `SendSilence` containers

## AutoCommit

Shared Objects are automatically committed when they have been changed.

### Description

Setting this tag to `false` disables the Flash Player function for all shared objects within this instance.

| NOTE | If the `AutoCommit` function is disabled, the server-side script has to call the `save` function or the `SharedObject.commit` command for the shared object to persist; otherwise, all data will be lost when the application is shut down. |
|---|---|

## Bandwidth

Container tag.

### Description

The tags nested in this section specify the bandwidth settings for upstream (client-to-server) and downstream (server-to-client) data.

By default, the `Bandwidth` tag includes an `override` parameter set to "`yes`", which allows the values for the `ClientToServer` and `ServerToClient` tags nested in this section to be overridden too.

### Contained tags

`ClientToServer (Bandwidth)`, `ServerToClient (Bandwidth)`

### See also

`BandwidthCap`

## BandwidthCap

Container tag.

### Description

The tags in this section specifies the bandwidth settings that a user can set. By default, this tag includes an `override` parameter set to "`yes`", which allows the values for the `ClientToServer` and `ServerToClient` tags nested in this section to be overridden too.

### Contained tags

`ClientToServer (BandwidthCap)`, `ServerToClient (BandwidthCap)`

### See also

Bandwidth

## Bits

This tag contains the settings for Flash Player on the Windows and Macintosh platforms.

### Examples

```
<Bits from="WIN 6,0,0,0" to="WIN 7,0,55,0">0x01</Bits>
<Bits from="MAC 6,0,0,0" to="MAC 7,0,55,0">0x01</Bits>
```

### See also

UserAgent

## CachePrefix

This tag specifies the cache prefix that is passed from the origin server to the proxy server.

### Description

This tag is set on the origin server. The proxy or edge server uses the value of this tag as a relative path to locate the cache file defined in the `CacheDir` tag.

The `type` attribute provides additional specification for the cache prefix. The `type` attribute can be set to `path` or `sname`. The default is `path`.

### Examples

```
<CachePrefix type="path"></CachePrefix>
```

When the attribute `type is "path"`, Flash Media Server appends the physical path of the recorded stream to the prefix.

```
<CachePrefix type="sname"></CachePrefix>
```

When the attribute `type = "sname"`, Flash Media Server appends the stream name to the prefix.

The cache prefix is any text with or without preset parameters. The prefix can be any name without special characters such as \, :, *, ?, ", <, >, |. All parameters are surrounded by ?. Flash Media Server will substitute the actual names for everything specified within the ?.

By default, the prefix is set to ?IP?

| Cache prefix | Actual name |
|---|---|
| `?IP?` | IP address of the server |
| `?APP?` | Application name |
| `?APPINST?` | Application instance |
| `?VHOST?` | vhost name |

You can include the IP address in the prefix to avoid file collision. For example, the proxy server might be connecting to two different origin servers with the same file in c:\data\foo.flv. Adding the IP to the prefix for these files points each file to the appropriate server.

If you want more than one origin server to share the cache file, do not include the IP as a parameter. Remember the cache prefix is a relative path used by proxy server to look up the cache stream file.

### Examples

The cache prefix creates a relative path in the proxy's `CacheDir`. All parameters are separated by '/' or '/'.

```
<CachePrefix type="path">c:\fms\flvs\foo.flv. data/?IP?</CacheDir>
```

resolves to:

```
data/xxx.xxx.xxx.xxx/c/fms/flvs/foo.flv
```

```
<CachePrefix type="path">?APPINST?/data</CacheDir>
```

resolves to:

```
app1/inst1/data/c/fms/flvs/foo.flv
```

```
<CachePrefix type="path">origin1/data/</CacheDir>
```

resolves to:

```
origin1/data/c/fms/flvs/foo.flv
```

## CacheUpdateInterval

This tag defines the wait interval for updating the cache streaming in the proxy server.

### Description

The interval is defined in milliseconds. The default value is 10 minutes. The minimum interval is 10 seconds. The maximum interval is 24 hours.

## Client

Container tag.

### Description

The tags nested within this container configure the client.

### Description

By default, the `Client` tag includes an `override="no"` parameter. Individual applications cannot override how the tags in the `Client` section are configured.

### Contained tags

`Access`, `Bandwidth`, `BandwidthCap`, `HTTPTunnel`, `UserAgent`

## ClientToServer (Bandwidth)

This is one of two tags named `ClientToServer` in the Application.xml file.

Located in `Bandwidth` container.

### Description

This tag specifies the maximum bandwidth the client can use for sending data upstream to the server. The default bandwidth is 250000 bytes per second.

### See also

`ServerToClient (Bandwidth)` in this container; `ServerToClient (Bandwidth)`, `ClientToServer (BandwidthCap)` in the `BandwidthCap` container.

## ClientToServer (BandwidthCap)

This is one of two tags named `ClientToServer` in the Application.xml file.

Located in `BandwidthCap` container.

### Description

This tag specifies the maximum bandwidth a user can set for data to be sent upstream from the client to the server. The default bandwidth is 10,000,000 bytes per second.

### See also

`ServerToClient (Bandwidth)` in this container; `ServerToClient (BandwidthCap)`, `ClientToServer (Bandwidth)` in the `Bandwidth` container

## CombineSamples

Container tag.

### Description

Flash Media Server conserves system resources by combining sound samples. This strategy saves the CPU and bandwidth overhead when transmitting individual audio packets only.

| NOTE | Use this strategy of combining sound sample advisedly during periods of high CPU usage as it can induce latency. |
|---|---|

### Contained tags

`LoCPU`, `HiCPU`, `MaxSamples`, `Subscribers`

## Connections

Container tag.

### Description

The tags in this section configure the HTTP connections for this application.

### Contained tags

`Interface`, `MaxTimeOut (Connections)`, `Reuse`

## DuplicateDir (SharedObjManager)

This is one of two tags named `DuplicateDir` in the Application.xml file.

Located in `SharedObjManager` container.

This tag specifies the physical location where duplicate copies of shared objects are stored.

### Description

This location serves as a backup for shared object files. This location must already exist and when a shared object is copied here it will be categorized by instance name by default.

### Example

```
<DuplicateDir appName="true">c:\backupSharedObjects</DuplicateDir>
```

To include the application name in the paths for the backup files, change the `appName` attribute to `"true"`.

### See also

`StorageDir (SharedObjManager)`

## DuplicateDir (StreamsManager)

This is one of two tags named `DuplicateDir` in the Application.xml file.

Located in `StreamManager` container.

This tag specifies the physical location where copies of recorded stream files are stored.

### Description

This location serves as a backup for stream files.  This location must already exist before a stream can be stored. By default, when a stream is copied to this location, it is categorized by instance name.

### Example

```
<DuplicateDir appName="true">c:\backupStreams</DuplicateDir>
```

To include the application name in the path for the backup files, change the `appName` attribute to `"true"`.

### See also

`StorageDir (StreamManager)`

## Duration

This tag instructs Flash Media Server how long to wait before it notifies the client when the audio has stopped in the middle of a live or recorded audio stream.

### Description

The default wait time is 3 seconds. The minimum wait time is 1 second.

## EnhancedSeek

This tag enables or disables fine tuning the seeking performance within streams by creating a keyframe.

### Description

Keyframes improve the visual display of FLV files while seeking. When this tag is set to `true`, the server inserts keyframes at the point in the stream where the seek begins if there is no preexisting keyframe present.

By default, this tag is set to `false`. The server does not insert keyframes and all seeks begin at the nearest existing keyframe.

### See also

`KeyFrameInterval`

## FileObject

Container tag.

### Description

The `VirtualDirectory` tag nested within this container configures the `JSEngine` file object settings.

### Contained tags

`VirtualDirectory`

## FolderAccess

This tag configures folder-level permissions for the `readAccess` and `writeAccess` functions in the Access Module.

### Description

By default, folder-level permission in the Access Module is set to false, which allows access permissions to be set at the single-file level. When the value of this tag is set to `true`, you cannot configure individual files for read or write access.

### See also

`Access`

## HiCPU

This tag instructs Flash Media Server to start combining samples when the CPU utilization is higher than the specified percentage of the CPU resource.

### Description

Default percentage of utilization is 80.

### See also

`LoCPU`

## Host

This tag identifies the HTTP Proxy.

### Description

The value for the `Host` tag can be the host name or a dotted IP address.

### Example

```
<Host>myserver:8080</Host>
```

To specify the port number in this string, add `:[port]` to the end of the host name.

The port number can also be specified in the `Port` tag.

### See also

`Port`

## HTTP

Container tag.

### Description

The tags in this section configure the HTTP connection settings for this application.

### Contained tags

`Connections` and `Proxy` containers; `HTTP1_0` and `Verbose` tags

## HTTP1_0

This tag determines whether or not Flash Media Server can use the HTTP 1.0 protocol.

### Description

The default is `false`, disallowing the use of the HTTP 1.0 protocol.

## HTTPTunnel

Container tag.

### Description

The tags nested within this container configure the parameters for HTTP tunneling (sending RTMP packets through HTTP).

The tunneling protocol is based on the client continuously polling the server. The frequency of polling affects both network performance and the efficiency of the HTTP protocol. The `IdleAckInterval` and `IdlePostInterval` tags control the polling frequency on a per-client basis. Selecting too small a delay value for the above parameters will increase the polling frequency and reduce the network performance and efficiency. Selecting too high values can adversely affect the interactivity of the application and the server.

The Application.xml configuration file offers three representative settings for these parameters. These settings recommend that you set the intervals to correspond to low, medium, or high latency.

The following table presents these settings.

| Acceptable Latency | IdlePostInterval | IdleAckInterval |
|---|---|---|
| Low | 128 milliseconds | 256 milliseconds |
| Medium | 512 milliseconds | 512 milliseconds |
| High | 1024 milliseconds | 2048 milliseconds |

### Contained tags

IdleAckInterval, IdlePostInterval, MimeType, WriteBuffSize

## IdleAckInterval

This tag specifies the maximum time the server may wait before it sends back an ack (acknowledgement code) for an idle post sent by the client.

### Description

The server may respond sooner than the value of this tag if it has data to send back to the client or if some other client is being blocked by the current idle request.

This interval implies that the client may not be able to reach the server for the selected duration. The interval cannot be set to a negative value.

The default interval is 512 milliseconds.

### See also

HTTPTunnel, IdlePostInterval

## IdlePostInterval

This tag specifies how long Flash Player should wait before sending an idle post to the server.

### Description

Idle posts are sent when Flash Player has no data to send but posting is necessary to provide the server with an opportunity to send data downstream data to the client.

The interval for an idle post ranges from 0 to 4064 milliseconds. If the `IdlePostInterval` tag is set to a value that lies outside of this range, the default value of 512 milliseconds is used.

| NOTE | At times the server will not be able to send any data to the client for the selected duration. |
|---|---|

### See also

`HTTPTunnel`, `IdleAckInterval`

## Interface

This tag defines the name to use as the outgoing network interface.

### Description

The name can be an interface name, an IP address, or a host name.

## Interval

This tag specifies in milliseconds the interval for sending silence messages when no audio is being published to a live stream.

### Description

Silence messages are used to support older versions of Flash Player. Flash Media Server will only send the silence message to clients which are specified in the `UserAgent` tag in the `Client` section. Bit-flag 0x01 is used to control the silence message.

The default interval is 3 seconds. Set this to 0 to disable the silence message transmission.

### See also

`UserAgent`

## JSEngine

Container tag.

### Description

The tags nested within this container configure the JavaScript engine.

### Contained tags

`FileObject` container; `MaxTimeOut (JSEngine)`, `RuntimeSize`, and `ScriptLibPath` tags

## KeyFrameInterval

This tag defines how often to generate and save keyframes in an FLV file.

### Description

Setting this tag to a higher value than the default reduces the number of keyframes added to the FLV file and thus reduces the file size. Setting a higher value for the interval, however, reduces the seeking accuracy. The value for this tag is defined in milliseconds. The default value is 1000.

For example, a 15-second video with a file size of 76 KB is increased only to 89 KB when the `KeyFrameInterval` tag is set to 5000, which is an increase of 13 KB, or 17%. The same video has a size of 109 KB with the `KeyFrameInterval` tag set to 1000, which is an increase of 33 KB, or 43%.

| NOTE | Be aware of the correlation between file size and accuracy of seeking when you set this value. |
|---|---|

### See also

`EnhancedSeek`

## LifeTime

Container tag.

### Description

This tag determines the lifetime of stateless core processes. To roll over such processes, set this tag to a non-zero value.

Process rollover happens only when the `Scope` tag is set to `inst`.

### Contained tags

`MaxCores`, `RollOver`

### See also

`Scope`

## LoadOnStartup

This tag determines whether or not the Flash Media Server loads an application instance when the server starts.

### Description

Having an application instance loaded at server startup saves time when the first client connects to that application. The default value is `false`.

If you set this tag to `true`, an instance of each application on the server will be loaded at startup.

## LockTimeout

This tag specifies the timeout value before automatically unlocking a shared object if there is a client waiting for an update.

### Description

The time-out value is specified in seconds. The default value is -1,which instructs Flash Media Server to wait for an indefinite time.

## LoCPU

This tag instructs Flash Media Server to stop combining samples when the CPU utilization is lower than the specified percentage of the CPU resource.

### Description

Default percentage of utilization is 60.

### See also

`HiCPU`

## Max

This tag defines the maximum number of redirects allowed.

## MaxAppIdleTime

This tag specifies the maximum time an application instance can remain idle with no clients connected before it is unloaded from the server's memory.

### Description

An application instance is evaluated as idle after all clients disconnect from it. If the application instance is loaded with no clients connected, it is not evaluated as idle.

The maximum idle time is specified in seconds. The default is 20 minutes.

## MaxCores

The value for this tag determines how many core processes can exist for an application.

### Description

By default, the `MaxCores` functionality is disabled. The default value is zero.

### See also

`LifeTime`, `RollOver`

## MaxFailures

The value for this tag determines the maximum number of process failures that can occur before a core process is disabled.

### Description

Once the core processes are disabled, Flash Media Server does not launch a core process until some minimum recovery time has elapsed. Having a time lag for recovery avoids a Denial of Service action, which can happen when a faulty core consumes all CPU resources by repeatedly launching itself.

### See also

`LifeTime`, `RecoveryTime`

## MaxMessagesizeLosslessvideo

This tag specifies the maximum size of messages for screen-sharing packets.

## MaxSamples

This tag specifies how many sound samples can be combined into one message.

### Description

The default number of samples is 4.

## MaxTimeOut (Connections)

This is one of two tags named `MaxTimeOut` in the Application.xml file.

Located in the `Connections` container.

### Description

This tag defines the maximum time for a transfer to be completed. The default time is 60 seconds.

Operations such as DNS lookups may take more time. If the setting for this tag is set too low a value, the risk of aborting correctly functioning operations increases.

### See also

`MaxTimeOut (JSEngine)` in the `JSEngine` container

## MaxTimeOut (JSEngine)

This is one of two tags named `MaxTimeOut` in the Application.xml file.

Located in the `JSEngine` container.

### Description

This tag specifies in seconds the maximum time a JavaScript can take to execute a JavaScript function. If its execution takes longer than the maximum allowed time, then the script is evaluated as a runaway script and its execution is terminated. Setting a maximum time to execute a script prevents infinite looping in scripts.

The default value is 0 and no checks are performed to detect runaway scripts. This setting may be useful in a debugging environment. In a production environment, after the applications and scripts have been thoroughly tested, you should set this tag to a more realistic value that does not impose limits on the time scripts take to execute.

### See also

`MaxTimeOut (Connections)` in the `Connections` container

## MimeType

This tag specifies the default MIME (Multipurpose Internet Mail Extensions) type header sent on tunnel responses.

### Description

The server generally uses the MIME type specified by the incoming requests. The server will use the entry for the `MIMEType` tag only when it is unable to determine the MIME type from the incoming requests.

## NotifyAudioStop

Container tag.

### Description

The `Duration` tag nested within this container determines whether or not Flash Media Server is notified when an audio transmission ending on a stream is encountered.

### Example

```
<NotifyAudioStop enabled="false"></NotifyAudioStop>
```

### Contained tag

`Duration`

## Password

This tag specifies the password for connecting to the proxy.

### See also

`Username`

## Port

This tag specifies the proxy port to connect to if it is not specified as part of the host in the `Host` tag.

### See also

`Host`

## Process

Container tag.

### Description

The tags nested within this container configure this relationship and how a process is recovered.

### Description

In Flash Media Server, applications are associated with processes.

### Contained tags

`LifeTime`, `MaxFailures`, `RecoveryTime`, `Scope`

## Proxy

Container tag.

### Description

The tags nested within this container configure the HTTP Proxy settings.

### Contained tags

`Host`, `Password`, `Port`, `Tunnel`, `Type`, `Username`

## RecoveryTime

This tag specifies the recovery time for a core.

### Description

Flash Media Server will not launch a core process until some minimum recovery time has elapsed. The time lag for recovery can avoid a Denial of Service action, which happens when a faulty core consumes all CPU time by repeatedly launching itself.

The recovery time for a core process is specified in seconds. A value of 0 disables any checking for process failures.

| NOTE | Loading an application with the Flash Media Admin Service tools or APIs will bypass this check. |
|---|---|

### See also

`MaxFailures`

## Redirect

Container tag.

### Description

The tags nested within this container configure the settings for redirecting the HTTP connection.

### Contained tags

`Allow`, `Max`, `UnrestrictedAuth`

## ResyncDepth

This tag instructs Flash Media Server to resynchronize a shared object file.

### Description

The shared object is resynchronized when its version number is greater than the head version minus the current version.

The default value `s -1` sends a resynchronized version of the file with every connection.

## Reuse

This tag configures whether or not Flash Media Server explicitly closes the HTTP connection after each transfer.

### Description

The default is to reuse connections. Set this to false to use a new connection after every transfer.

## RollOver

This tag specifies how long a core process can be in use before Flash Media Server creates a new core process.

### Description

After the time limit for a core is reached, a new core is instantiated. All subsequent connections are directed to the new core.

The rollover functionality is disabled by default. The default value is 0.

### See also

`LifeTime`, `MaxCores`

## RuntimeSize

This tag specifies the maximum size in kilobytes that a particular application instance can use to run server-side ActionScript code before Flash Media Server removes unreferenced and unused JavaScript objects.

### Description

The default size is 1024 kilobytes, which is the equivalent of 1 megabyte. The lower and upper limits on the size of the JavaScript engine are 10 kilobytes and 51200 kilobytes, which is the equivalent of 50 megabytes. The default value applies when the engine size lies outside of these limits.

If your application consumes a significant amount of memory, you must increase the engine size. If you create a new script object that will cause the runtime size of the application instance to exceed the value of `this` tag, an out-of-memory error occurs and the application instance is shut down.

| CAUTION | Do not change the engine's size without technical assistance. |
|---|---|

## Scope

This tag determines the scope in which Flash Media Server runs this application.

### Description

Set this tag to `app` to run an application and all its instances as a single process, or to `inst` to run each instance in a separate process. The default setting is "`app`."

If no value is specified for this tag, each virtual host, its applications and application instances are run as a single process.

### See also

`LifeTime`, `MaxFailures`, `RecoveryTime`

## ScriptLibPath

This tag is a list of paths delimited by semicolons instructing Flash Media Server where to look for server-side scripts.

### Description

These paths are used to resolve a script file that is loaded with the load API. The server first looks in the location where the main.asc or *application_name*.asc file is located. If the script file not found there, the script engine searches in sequence the list of paths specified in this tag.

## SendSilence

Container tag.

### Description

The `Interval` tag nested within this container configures the settings for sending silent messages.

### Contained tag

`Interval`

## ServerToClient (Bandwidth)

This is one of two tags named `ServerToClient` in the Application.xml file.

Located in the `Bandwidth` container.

This tag specifies the maximum bandwidth the server can use for sending data downstream to the client.

### Description

The default bandwidth is 250000 bytes per second

### See also

`ClientToServer (Bandwidth)`, `ServerToClient (BandwidthCap)` in the `BandwidthCap` container

## ServerToClient (BandwidthCap)

This is one of two tags named `ServerToClient` in the Application.xml file.

Located in the `BandwidthCap` container.

This tag specifies the maximum bandwidth a user can set for data to be sent downstream from the server to the client.

### Description

The default bandwidth is 10,000,000 bytes per second.

### See also

`ClientToServer (BandwidthCap)`, `ServerToClient (Bandwidth)` in the `Bandwidth` container

## SharedObjManager

Container tag.

### Description

The tags nested within this container configure the Shared Object Manager setting of an application.

### Contained tags

`AutoCommit`, `DuplicateDir (StreamsManager)`, `LockTimeout`, `ResyncDepth`, `StorageDir (StreamManager)`

## StorageDir (SharedObjManager)

There are two tags named `StorageDir` in the Application.xml file; this one is in the `SharedObjManager` container.

This tag specifies the physical location where shared objects are stored.

### Description

By default the physical location is not set. Set this tag only if the files for shared objects must be stored in a location other than the application directory.

By default, this tag is not set as it remaps the location where files are stored. Set this tag only if you want to remap and store these files in a location other than the application directory.

### Example

```
<StorageDir>C:\myapp\sharedobjects\</StorageDir>
```

### See also

`DuplicateDir (SharedObjManager)`

## StorageDir (StreamManager)

There are two tags named `StorageDir` in the Application.xml file; this one is in the `StreamManager` container.

This tag specifies the physical location where recorded streams are stored.

### Description

By default the physical location is not set.

### Example

```
<StorageDir>C:\myapp\streams\</StorageDir>
```

Set this tag only when the files for recorded streams must be stored in a location other than the application directory.

### See also

`DuplicateDir (StreamsManager)`

## StreamManager

Container tag.

### Description

The tags in this section configure the Stream Manager settings for this application.

### Contained tags

`Audio` container

`CachePrefix`, `CacheUpdateInterval`, `DuplicateDir (StreamsManager)`, `EnhancedSeek`, `KeyFrameInterval`, `StorageDir (StreamManager)` tags

## Subscribers

This tag instructs Flash Media Server to combine sound samples only if there are more than the default number of subscribers to that stream.

### Description

The default number of subscribers is 8.

## Tunnel

This tag specifies whether or not to tunnel all operations through a given HTTP proxy.

### Description

The default setting is `false`.

## Type

This tag specifies the type of proxy being connected to.

### Description

The value for this tag can be `HTTP` or `SOCKS5`. The default is `HTTP`.

## UnrestrictedAuth

This tag determines whether or not to allow sending the username/password combination with each HTTP redirect.

### Description

Sending the username/password combination is useful only if the `Allow` tag permits redirections. The default setting is `true`.

## UserAgent

Container tag.

### Description

The settings for clients vary according to whether the Flash Player platform is Windows or Macintosh. Setting the value 0x01 will configure the player and platform for silence messages.

### Contained tag

`Bits`

### See also

`Interval`

## Username

This tag specifies the username for connecting to the proxy.

### See also

`Password`

## Verbose

This tag determines whether or not Flash Media Server outputs verbose information during HTTP operations.

## VirtualDirectory

This tag specifies the virtual directory mappings for file objects in a JavaScript.

### Description

Virtual directories lets you specify file directories for different applications. If the beginning portion of a file path matches the specified virtual directory, then the storage location of the file becomes the file path of the virtual directory.

### Syntax

```
<VirtualDirectory><virtual dir>;<actual dir></VirtualDirectory>
```

## WriteBuffSize

This tag specifies in kilobytes the size of the write buffer.

### Description

The default size is 16KB.

CHAPTER 4

# Flash Media Server Security

4

Macromedia Flash Media Server 2 will typically be used in a network environment where many users will have access to it; by changing its configuration, you can make the server accessible from within a private network, from the public Internet, or both. When deploying any server technology, you should consider the implications to both the security of your internal network and the accessibility of the server's host computer.

Flash Media Server incorporates security features that take these kinds of concerns into account. As a server administrator, you can provide additional security. This chapter describes the security features built into Flash Media Server as well as additional measures you can take to protect your server.

Additional information about server security can be found in the Macromedia Flash Media Server Support Center at www.macromedia.com/go/flashmediaserver_support_en.

## Managing server security

Flash Media Server uses a high-speed TCP/IP protocol called Real-Time Messaging Protocol (RTMP), which is binary and unencrypted. RTMP is sufficient for many media applications, such as those that run within your organization's intranet. For applications that handle critical or sensitive data, use the secure RTMPS protocol, which encrypts all data, including audio and video. For more information about using RTMPS in applications, see Access DLL.

Because the RTMP protocol is unencrypted, you must carefully consider the security of your server configuration and the sensitivity of the data you send to and from the server.

The default settings of Flash Media Server at installation provide sufficient security. As a server administrator, you can enhance the level of security by modifying the default settings in the configuration files. The following section identifies those configuration tags that reinforce security in Flash Media Server.

**Edit the security tags in the configuration files** Utilize the limits that can be set in the server's configuration files. Use the following tags in the configuration files to enhance the server's security:

- Server.xml file

  The `HostPort` tag nested in `AdminServer` container allows you to specify the port of your choice for connecting to the Admin service with the management console. This allows you to use a port that will work with your firewall configuration. The default is port 1111.

  The `ServerDomain` tag lets you specify the domain that the Flash Media Server is running in so that it can identify its domain to application servers you may want it to connect to.

  The `SSL` tags let you specify your digital certificates used for client connections. If you want secure connections using the RTMPS protocol, you must enter appropriate values in the `SSL` tags.

- Users.xml file

  The `User` tags allow you to specify exactly who can connect to the server with the management console. Only users specified with these tags can connect.

  The `Allow` and `Deny` tags let you specify exactly which domains administrators can connect from. Administrators cannot connect from domains that are not permitted with these tags. If you are running the server on a Linux system, remember to allow connections from the domains where administrators will use the console to manage and monitor the server and its running applications.

- Adaptor.xml file

  The `Allow` and `Deny` tags let you specify exactly which domains administrators can connect from. Administrators cannot connect from domains that are not permitted with these tags. These tags indicate permissions specifically for the adaptor. If you are running the server on a Linux system, remember to allow connections from the domains where administrators will use the console to manage and monitor the server and its running applications.

  The `HostPort` tag allows you to specify the port to use for client connections and specify if a port is secure or not. This lets you choose a port that works with your firewall configuration. The default for RTMP connections is port 1935. The `secure` attribute of the `HostPort` tag, set to a value of `true`, allows you to specify that the port uses secure FRTMP (RTMPS) for client connections. The default secure port is 443.

- Vhost.xml file

  The `MaxConnectionThreads` tag nested in the `ResourceLimits container` allows you to limit the number of threads to use for processing input/output requests. This prevents denial-of-service attacks from bringing down the server computer itself.

The `MaxAppInstances` tag nested in the `ResourceLimits container` lets you limit the number of application instances that can exist simultaneously on the virtual host. This can help prevent denial-of-service attacks. The default is -1, which allows unlimited application instances.

The `MaxStreams` tag nested in the `ResourceLimits container` lets you specify the maximum number of streams that can exist simultaneously on the virtual host. This can help prevent denial-of-service attacks. The default is -1, which allows unlimited streams.

The `MaxSharedObjects` tag nested in the `ResourceLimits container` lets you specify the maximum number of shared objects that can exist simultaneously on the virtual host. This can help prevent denial-of-service attacks. The default is -1, which allows unlimited shared objects.

- Application.xml file

  The `RuntimeSize` tag nested within the `JSEngine` container lets you limit the amount of memory that can be used by the server-side ActionScript on the virtual host. This can help prevent attacks using very large numbers of scripts. The default is 1024K.

  The `StreamManager` and `SharedObjManager` tags nested in the `StreamManager` container let you specify the locations for storing streams and shared objects. You can store them in locations outside the applications directory in the Macromedia Flash Media Server directory, if you wish.

  The `Bandwidth` tags groups let you specify the maximum amount of data that an application can send and receive.

For detailed information about the server's configuration files, see Chapter 3, "Configuration Files."

**Place source and data files carefully** To prevent hackers from gaining access to the source files of your applications, do not place sensitive files in your web server's publishing directory. If you have a web server, the management console (fmsconsole.swf) is installed by default in your web server's publishing directory. During deployment, do not place Flash Media Server application source or data files (FLA, FLV, ASC) or the applications directory (installed by default in the Flash Media Server directory) in the web publishing directory; keep only your SWF and HTML files in the publishing directory.

**Protect configuration files** In addition to its media streams, the server's configuration files should be protected. To ensure that the server's configuration files and directory structure cannot be accessed by unauthorized users, place the server computer in a physically secure location and password-protect the operating system so that only the appropriate server administrators have access.

# About authentication and authorization

To authenticate (validate) administrators, Flash Media Server employs several layers of host-based user security. (*Host-based security* refers to security measures that are implemented in the server software itself.) When a user tries to connect to the management console with an administrator user name and password, the server uses the layers of settings in its configuration files to determine whether the connection should be allowed. Only administrators who have been explicitly defined can connect to the server to use the console.

The server authenticates administrators by evaluating the contents of the XML tags in the configuration files in the following order:

1. Users.xml file: `Allow` and `Deny` tags. These tags indicate whether a user is allowed to connect to the console from the current IP address. Administrators can connect only from IP addresses you have specified with these tags.
2. Adaptor.xml file: `Allow` and `Deny` tags. These tags indicate whether a user is allowed to connect to the specified adaptor from the current IP address.
3. Vhost.xml file: `Allow` and `Deny` tags. These tags indicate whether a user is allowed to connect to the specified virtual host from the current IP address.

The server authenticates administrators by comparing their user names and passwords to those defined in the Users.xml file. When you choose these names and passwords, make sure they are not simple ones that can be easily guessed.

To have the server perform authentication of connecting users other than administrators, use the `Allow` and `Deny` tags in the Adaptor.xml and Vhost.xml files. With these tags you can prevent users from connecting from all domains other than those you specify. The server checks incoming connections against the Adaptor.xml file and then the Vhost.xml file when processing non-administrator connection requests.

To provide administrator *authorization* (assigning permissions), the server uses the Users.xml file. When you define a user as a server or virtual host administrator in this file, the server associates certain permissions with that user. Virtual host administrators can manage only a virtual host—for example, they can reload or disconnect applications on that virtual host. Server administrators can exercise control over all virtual hosts and perform server-level tasks, such as restarting or shutting down the server.

By default, only the management console performs user authorization. When developing your own media applications, you can decide whether to implement user authorization; some kinds of applications need this capability while others do not. For example, when developing a simple chat application, you might choose to create two different versions of your Macromedia Flash client application. One version might be a chat participant version; another might be a chat moderator version, with additional functionality built in, such as the ability to edit users' posts or disconnect users. Using server-side ActionScript, you can define which users are able to connect with the moderator version of the application (SWF file).

As an additional security feature, the management console actually connects to the Flash Admin Service, which then communicates with the server service to perform administration tasks.

# JavaScript security

This release of Flash Media Server adds support for custom third-party pods. Pods are essentially a combination of user interface elements that, along with client and server-side ActionScript code, extend the features and functionality of Flash Media Server. There are security implications in using pods, since the pod code is not as tightly controlled as the main Flash Media Server application.

Flash Media Server enforces script security when using pods by limiting its execution only to the application into which it is included and by ensuring that its code does not adversely affect or compromise the content of other applications.

JavaScript security support consists of secure script loading and protected objects. Script code is loaded before the main application. This strategy hides user-defined objects behind restrictive C wrapper objects, which protects their methods and data from being inspected or manipulated. An application developer can implement system calls to protect critical data and functions, such as the built-in `global load()` and `setVirtualPath()` functions.

# Secure script loading

The Flash Media Server script security model enables one to limit the exposure to potentially malicious or buggy third-party code that may be included on the server side. An example would be an extensible application where users could download third-party plug-ins or components, then load or evaluate them in the application. If you are concerned that such plug-ins or components may compromise the system, you can apply the script security model to restrict them. The script security model is not designed to detect or prevent error conditions such as an infinite loop in third-party code, but it is useful for preventing or limiting certain potentially dangerous functionality such as the ability to make arbitrary connections, and read/write file objects.

Script security is probably not applicable for most applications, but it can be very valuable to anyone building dynamically extensible applications—the kind that loads and evaluates code from external sources.

When an application is started, it first looks for and loads the file secure.asc. During this period of time, it makes the APIs `protectObject()` and `getGlobals()` available. These may be used to manipulate global functions, classes and objects in a way that is not possible during normal application execution. Once Flash Media Server is done loading `secure.asc`, it makes these APIs unavailable. It then proceeds to load `main.asc` and other scripts in the normal manner.

For example, if you wanted to implement an ID generator that must generate an ever increasing numbers for IDs, then you would add a function like the following in your `main.asc` script.

### Example

```
idGen = {};
idGen._nextID = 0;
idGen.nextID = function() { return this._nextID++; }
```

This example suffices for generating IDs. However any part of the script can easily redefine the `nextID()` function or directly modify the `_nextID` value. There was no way to prevent redefinition in the previous versions of Flash Media Server. With the script security model, however, you simply add the code for the generator into `secure.asc`.

```
// Begin secure.asc
trace( "loading secure.asc" ) ;

var global = getGlobal(); // grab the global object
var idgen = {};
idgen._nextID = 0;
idgen.nextID = function() { return this._nextID++; }

// Create a protected object out of idgen and make it
```

```
// available globally as idGen.

global.idGen = protectObject( idgen );

// Make idGen non-enumerable, read-only and permanent
setAttributes( global, "idGen", false, true, true );
```

When normal script loading begins `idGen` will be available as a global object, that cannot be compromised by any script loaded directly or indirectly from `main.asc`.

### Example

```
//main.asc
trace( "Loading main.asc" );
trace( "idGen = " + idGen );
idGen = 50;
trace( "idGen = " + idGen );
```

Here's the output for main.asc:

### Example

```
Loading secure.asc
Loading main.asc
idGen = [object Redirector]
idGen = [object Redirector]
```

If you are using Linux, remember that `secure.asc` is case sensitive.

## Protecting objects

A new global function in Flash Media Server lets application developers protect user-defined objects behind C-wrapper objects. The function `protObj = protectObject(userObj);` takes an object and returns the wrapper object. Any user-defined object that has been protected with this `proObj` function becomes a wrapper object whose methods may be considered as system calls since they cannot be compromised.

The wrapper object returned by this function fulfills all invocations to the underlying user object but blocks access to the member data. As a result, one cannot enumerate or modify members directly. Once an object has been protected with this function, you need to make sure that it is no longer accessible in global variables or as a member of an accessible object.

The wrapper object keeps a reference to the underlying user object to ensure that it remains valid during the lifetime of the wrapper. The wrapper itself follows the normal reference rules and exists as long as an application refers to it.

Protected objects can be used to implement any security model, such as system calls, privilege rings, and ACLs (Access Control Lists).

# Permissions levels

Flash Media Server does not use explicit levels of privileges, but provides a way for the application developer to implement system objects that the application code can not compromise. Privileged access is simply the capability to directly access these special objects. These system objects could be compromised if a system call explicitly evaluates randomly accessed code on the caller's behalf. This should never be permitted.

## Synchronous system calls

The protected object mechanism may be used to simulate system calls. The system introduces a C-layer shim for each protected object to intercept the call and pass it to the underlying user-defined object. Application code can never directly access or inspect how Flash Media Server implements system objects. Developers can use this mechanism to disguise global functions by renaming and storing them as a protected object, which will make it available only through a wrapper. This technique for creating protected objects allows application developers to hide built-in global functions or implement new global functions.

The following example demonstrates how a `secure.js` script protects the `load` function.

```
      //Begin system.asc
    var sysobj = {};
sysobj._load = load;
  //Stash away the load function.
   load = null;
     //Make it unavailable unprivileged code.
   sysobj.load = function(fname) {
     //User defined code to validate/modify fname ...
        return this._load(fname);
   }
   // Grab the global object.
   var global = getGlobal();
   // Now protect our sysobj and make it available as
   // 'system' globally. Furthermore set its attributes
   // such that it is read-only and not deletable.
   global["system"] = protectObject(sysobj);
   setAttributes( global, "system", false, true, true );
   // Now add a global load() function for compatibility.
   // Make it read-only and non-deletable.
   global["load"] = function(path) { return system.load(path); }
   setAttributes(global, "load", false, true, true);
   // End system.asc
```

The act of an application calling the `load` function is always performed by the user-defined system call.

## Asynchronous system calls

In Flash Media Server, application developers can implement asynchronous system calls, where the caller is unprivileged and relies on a system call to set up and complete the call. The callback must remain unprivileged. This coding is useful when a system object is trying to wrap and hide a network connection.

```
// in secure.asc ...
sysobj.remoteCall = function(func, responder, arg1, arg2, ...) {
     // validate/modify args ...

     var sysResponder = {};
     sysResponder.sysobj = this;
     sysResponder.userResponder = responder;
     sysResponder.onResult = function(res) {
          // Modify/validate res ...
          // Perform any other 'privileged' funcs ...
          // Remove any access to system object.
          this.sysobj = null; delete this.sysobj;
          // Pass on the result to the user callback.
          this.userResponder.onResult(res);
     }
     this._nc.call(func, sysResponder, arg1, arg2, ...);
}
...
system = protectObject( sysobj );
...
```

The call would be invoked from the normal application code as:

```
system.remoteCall("foo", myOnResult, arg1, arg2);
```

In Flash Media Server, an asynchronous call triggered by the application code is never defined as privileged. The following example shows how application developers can ensure that the asynchronous calls set within the system object can be completed as privileged or unprivileged.

```
// within some system call implementation ...
var resultObj = {};
resultObj.sysobj = this;
resultObj.onResult = function(res) {
//use the stashed sys obj to do
// some privileged action ...
}
this._nc.call("foo", resultObj, ...);
```

# Choosing passwords

When choosing passwords, remember to make them as secure as possible. The following guidelines can help you create more secure passwords:

- The minimum length of a password should be 7 characters.
- Passwords should not contain your user name or any part of it (for example: Jane, Doe, Jdoe).
- Passwords should contain three of the following four items: at least one uppercase letter (A-Z), at least one lowercase letter (a-z), at least one numeric character (0-9), and at least one non-alphanumeric character shown here:

  ! _ * # $ % & ; + -
- Passwords should be changed regularly and none of the last five passwords should be reused.

# Access DLL

Access DLL is a new module in this release of Flash Media Server. This module adds another layer of security before Flash Media Server accepts connection requests from clients. Access DLL intercepts and examines each connection request to determine whether Flash Media Server should accept or reject the request. You can configure the Access DLL module to control settings of the client connection or to access relevant server statistics such as the number of connections. The Access DLL module is initialized upon Flash Media Server startup.

Flash Media Server administrators can deploy the Access DLL module as a programmatic method to intercept connections to Flash Media Server before the requests reach its JavaScript layer. The module can be configured to initiate a query of the organization's database of users and passwords to determine if a connection should be allowed, and if it is allowed, the connection is accepted and the database updated with a record of the user's access to Flash Media Server.

The module's `getStats` function reveals how many current connections exist and the current values for the `bytesIn` and `bytesOut` APIs. You can configure Access DLL to reject or accept users based upon how many users are currently connected to the server and the bandwidth being currently consumed. The client will poll these values when a connection is attempted. If the values received are below the defined threshold, then the connection is accepted. If the values defined for the `bytesIn` and `bytesOut` have been exceeded, the connection is rejected.

Access DLL is the `libconnect.dll` file (this module is named the `libconnect.so` file in Linux installations) stored within the modules/access subdirectory of the root Flash Media Server installation.

When a connection is attempted, Flash Media Server first determines whether or not Access DLL exists.

- If the Access module is available, the module is initialized on server startup with a context pointer. The module also provides an adaptor interface to the server. Together, the context pointer and adaptor pointer provide for two-way communications between the access module and the main server. This context pointer can be used to gather server statistics.
- If the Access module is not available, all connection requests proceed as usual. The connection is accepted pending resources and valid application name.

## Configuring Access DLL

The Flash Media Server administrator can configure how the Access DLL module handles connection requests for the server's resources. When the connection request from a client is first attempted, the Access module intercepts the request before sending it to the server. The Access DLL module calls upon its APIs to examine the information contained within the connection request and authorizes or rejects the connection request.

The Access module uses the following APIs to examine the request:

- The connection request activates the `onAccess` callback function within the Access module.
- The `setValue` function determines which field to examine and authenticate.
- You select this field when you configure the `setValue` API.
- The Access DLL module makes the value defined by the `setValue` function available to the `getValue` API.
- The `getValue` function returns the value found in the predefined field of the connection message.
- The `OnAccess` function accepts or rejects the connection.
- The `getStats` function collects server statistics from the context pointer.

When you configure this module, you can use the sample DLL authentication supplied by Macromedia or apply your own authentication mechanism to intercept and examine connection attempts to Flash Media Server.

# Access DLL APIs

Access DLL provides the following `AccessAdaptor` APIs:

| API name | Description |
|---|---|
| `getVersion` | Returns the version of the Access module. |
| `getDescription` | Returns a description of the Access module. |
| `onAccess` | Callback; this API is activated when a connection is attempted to Flash Media Server. OnAccess is responsible for accepting or rejecting a connection. |

Access DLL provides the following `AccessContext` APIs:

| API name | Description |
|---|---|
| `getVersion` | Returns the version of Flash Media Server to the Access module. |
| `getDescription` | Returns a description of the Access module. |
| `getStats` | Returns a selected server statistic such as number connected. |

Access DLL provides the following Access APIs. These APIs represent an individual access session. Both the Access module and the server can access these APIs.

| API name | Description |
|---|---|
| `getType` | Returns the type of the access event. |
| `getValue` | Returns one of the specific values associated with the connection. This API is configured to return the value of *one* of the following fields: c-referrer, c-user-agent, s-uri, or c-ip |
| `setValue` | Places a value into one of those values available to the `getValue` API. The `setValue` API can change the connection properties when the connection is passed to Flash Media Server.<br>Note that you cannot set the c-uri field. |
| `accept` | Allows the current connection being processed to connect to Flash Media Server. |
| `reject` | Denies a connection for the current session to Flash Media Server. The `reject` API also displays an explanation as to why the connection was denied. |

| API name | Description |
|---|---|
| `setReadAccess` | Sets the read access for a client. The Access string is configured as JavaScript's client.readAccess. The second parameter is a Boolean value with its default as `true`. This Boolean value, if `true`, will block user scripts from changing this value. If `false`, user scripts will be allowed to change this value. |
| `setWriteAccess` | Sets the write access for a client. The Access string is configured as JavaScript's client.writeAccess. The second parameter is a Boolean value with its default as `true`. This Boolean value, if `true`, will lock user scripts from changing this value. If `false`, user scripts will be allowed to change this value. |

# Modifying Access DLL

The Access DLL module contains the Sample.dsp file. You can change or configure this file to conform to your site-specific conditions and needs for authenticating connection requests to Flash Media Server.

For an excerpt of this file, see "Sample Adaptor.cpp file" on page 230.

| File name | Description |
|---|---|
| `Adaptor.cpp` | C++ file that contains the code for Access DLL. |
| `Adaptor.h` | Header file associated with Adaptor.cpp. |
| `Makefile.access` | Work file to be used for building Access DLL in a Linux environment. |
| `Readme.txt` | Text file with more information on Access DLL. |
| `Sample.dsp` | Work file to be used with Microsoft Visual C++ for building Access DLL in a Windows environment. You can change or modify the values to reflect the authentication practices in your environment. |
| `Sample.dsw` | Work file to be used with Microsoft Visual C++ for building Access DLL in a Windows environment. You can change or modify the values to reflect the authentication practices in your environment. |

## Sample Adaptor.cpp file

Here is an excerpt from the Adaptor.cpp file that you can modify to fit your local authentication profile. Adaptor.cpp is a C++ file that contains the code for the Access DLL module.

```
The corresponding file on Linux systems is called Makefile.access.
void SampleAdaptor::onAccess(IFCAccess* pAccess)
{
            switch(pAccess->getType())
{
              case IFCAccess::CONNECT:
              {
                     fprintf( stderr, "SampleAdaptor [Connect] referrer:
  %s\n", safestr(pAccess->getValue("c-referrer")) );
                     fprintf( stderr, "SampleAdaptor [Connect] uri: %s\n",
  safestr(pAccess->getValue("s-uri")) );
                     fprintf( stderr, "SampleAdaptor [Connect] user agent:
  %s\n", safestr(pAccess->getValue("c-user-agent")) );
                     fprintf( stderr, "SampleAdaptor [Connect] client ip:
  %s\n", safestr(pAccess->getValue("c-ip")) );

                     char *strValue = (char
  *)malloc(STRING_VALUE_BUFFER_LEN);
                     memset((void *)strValue, 0, STRING_VALUE_BUFFER_LEN);

                     m_pCtx->getStats("s-connected", strValue);
                     fprintf( stderr, "SampleAdaptor [Connect] getStats,
  connected %s\n", safestr(strValue) );

                     m_pCtx->getStats("s-bytes-in", strValue);
                     fprintf( stderr, "SampleAdaptor [Connect] getStats,
  bytes-in %s\n", safestr(strValue) );
                     m_pCtx->getStats("s-bytes-out", strValue);
                     fprintf( stderr, "SampleAdaptor [Connect] getStats,
  bytes-out %s\n", safestr(strValue) );

                     // Reset the uri string.
                     pAccess->setValue("s-uri", "rtmp://this.somewhere-
  else.com");
                     //fprintf( stderr, "Adjusted [Connect] uri: %s\n",
  safestr(pAccess->getValue("s-uri")) );
                     pAccess->setValue("c-referrer", "new referrer");
                     pAccess->setValue("c-user-agent", "other user agent");

                     pAccess->setReadAccess("/", false);
                     pAccess->setWriteAccess("/", true);

                     break;
```

```
}

                    default:
                    // We really shouldn't get here!
                    fprintf( stderr, "SampleAdaptor: Unknown access
  event!\n" );
}

//pAccess->reject("why not");
pAccess->accept();
}
```

# Developing secure applications

If you develop Flash Media Server applications, you can use SSL (Secure Sockets Layer) and other secure development practices to ensure the security of your applications and the data they use.

## Using SSL

To use SSL in your applications, you need to configure both your applications and Flash Media Server settings. The following list is a checklist of the steps you need to take to use SSL.

**Use RTMPS to connect to the server.** RTMPS adheres to SSL standards for secure network connections. It offers basic connectivity through a TCP socket on a secure port. Data passed over a secure connection is encrypted to avoid eavesdropping by unauthorized third parties. Because secure connections require extra processing power and might affect the server's performance, use RTMPS only for applications that require a higher level of security or that handle sensitive or critical data.

To use RTMPS, see the NetConnection.connect entries in the *Client-Side ActionScript Language Reference for Flash Media Server 2* and *Server-Side ActionScript Language Reference*. If, in your NetConnection.connect call, you don't specify a port number, Flash Player connects on the default secure port, port 443. Be sure that either port 443 or another designated port number is specified as secure using the `secure` attribute in the `<HostPort>` tag of the Adaptor.xml file.

| TIP | You cannot use RTMPS to establish a connection from one server running Flash Media Server to another if the server being connected to is behind a firewall that is rejecting RTMP data. A workaround is to cluster all servers running Flash Media Server behind the same firewall. |
|---|---|

**Configure the adaptor to listen on a secure port.** If you need a secure connection, configure the adaptor for the application to listen on a secure port by setting the `secure` attribute to `true` in the `HostPort` tag in the Adaptor.xml file. Be aware that you can assign only one virtual host to an adaptor that listens on a secure port, and you must specify the IP address of that virtual host in Adaptor.xml.

| TIP | If a secure connection is not required, you can use assign more than one virtual host to a single adaptor; that is, you can have multiple virtual hosts on a single IP address. |
|---|---|

**Configure adaptors globally or individually.** The `SSL` section in the Server.xml file contains required information for using SSL and configures all adaptors to use the same settings. However, you might want to use a different certificate for each virtual host. For example, if you are an Internet service provider, you might want different websites that you host to send different digital certificate information to clients. In this case, you would configure your adaptors individually to override the settings in the Server.xml file.

To use different certificates for each adaptor, copy the `SSL` section in the Server.xml file to the Adaptor.xml file and enter the new values. You don't need to copy the `SSLRandomSeek` tag, as this tag is a server-level setting that cannot be overridden in Adaptor.xml.

## Using other secure development practices

You might not want to use SSL in all your applications because of the additional processing time required to encrypt data over a secure connection. You can use other effective strategies to help protect all your media applications, regardless of what protocol is used for connections.

**Confirm the location of the client SWF** When you deploy a Flash Media Server application, use a server-side script to verify that connecting SWF files are coming from the location you expect (and not from an unknown computer). You can do this by checking the `client.referrer` property of the client object before the server accepts the connection.

For more information about writing server-side scripts, see *Developing Media Applications*.

**Use server-side script precautions** In server-side scripts do not use procedures that can be called by a malicious application, which could then fill a hard disk, consume the processor, or do other damage. Procedures attached to client objects are particularly vulnerable. Procedures to be aware of include writing to the hard disk without checking the quantity of data being written, procedures that can be infinitely looped, and so on.

**Send sensitive data via HTTPS** If you need to send sensitive data such as credit card information, you can use HTTPS to communicate simultaneously between your Flash client application and a separate application server that processes the data. To do this, use the ActionScript getURL command. (For more information, see the *ActionScript 2.0 Language Reference*.)

# About privacy

The technology in Flash Media Server enables the capture of client audio and video streams. When creating applications, it is your responsibility to comply with all applicable laws, rules, and regulations and to inform the user of privacy rights and your policies in situations such as when the application transports audio or video data across insecure channels or when audio or video data is being recorded for publication.

## Deploying secure applications

When you deploy a Flash Media Server application, it is important to take steps to ensure that your network is secure.

In addition to the precautions taken during the application development process, you should deploy your media applications in a firewall-protected environment. Firewalls provide port-based protection for your network and can be used to prevent connections to the network from specific IP addresses.

You should take precautions when using log files to track server activity, since these files can consume large amounts of disk space over time.

The following two sections describe these precautions in more detail.

## About firewalls

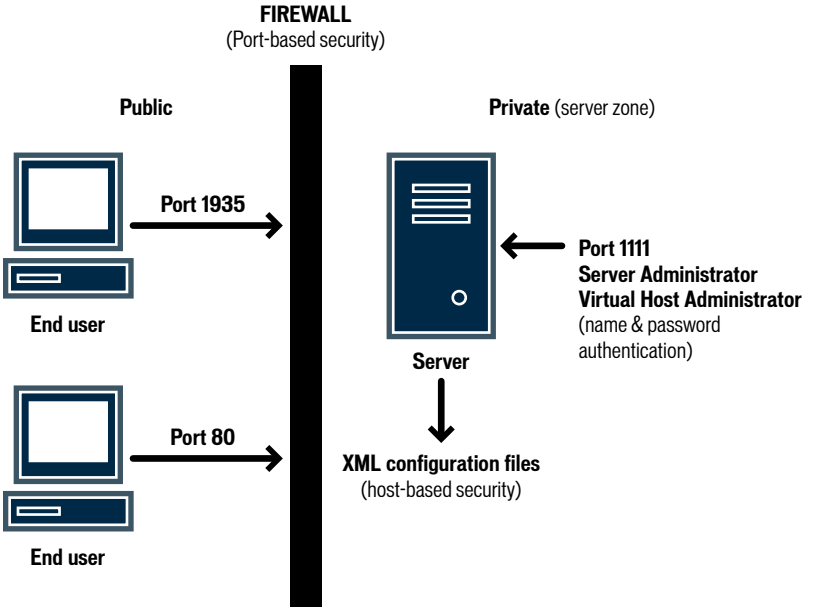
A *firewall* is a combination of hardware and software that controls the flow of information between networks, such as between a company intranet and the wider Internet. Firewalls provide port-based security, meaning they can be configured to allow certain media ports (1935, 80) to appear "outside" the firewall, making them accessible to external networks.

The port that Flash Media Server uses should be behind a firewall if it's being used only by users of a private network, such as a corporate intranet. The port should be accessible from outside the firewall if it's meant to be accessible to outside users such as users of the Internet in general.

If the Flash Media Server and an application server are both behind a firewall, they can communicate with each other and no outside party can eavesdrop on the data to gain access to private information.

You can also configure a firewall to provide additional protection against outside attacks. For example, if the server is being flooded by a particular IP or range of IP addresses, you can configure the firewall to ignore messages from those IP addresses.

The server allows you to strictly control which users can connect to it and where they can connect from. You can also configure a firewall to control the ports users inside and outside your network can connect to.




## Log file precautions

A log file is a file that contains information about events that have occurred on the server. When using log files, you are vulnerable to denial-of-service attacks by applications that can fill the hard disk—for example, by making high volumes of connection requests. To prevent this problem, write an operating system script to delete or back up the log regularly. You can also configure the duration and rotation of these files in the Logger.xml file.

# Index

## A

## C

## D



## E



## F



## H



## J



## K



## L



## M

## O



## P



## R



## S

## T



## U

## V



## W